OBJECT, ACTION, AND OUTCOME BLENDING LATENT SPACE EXPLORATION WITH INTRINSIC MOTIVATION TO LEARN MANIPULATION SKILLS

by

Melisa İdil Şener

B.S., Computer Engineering, Middle East Technical University, 2017

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2020

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor Assist. Prof. Emre Uğur for his guidance, encouragement, and motivation that helped me throughout my master's studies. My sincere thanks to Prof. Erhan Öztop and Dr. Yukie Nagai for their contribution to my research. I wish to thank Assoc. Prof. Hatice Köse and Assist. Prof. İnci Ayhan, for kindly accepting to be a member of my thesis jury.

My sincere thanks to Serkan Buğur for his friendship, for our conversations about cool stuff, for fruitful discussions about research, and for making everything fun. I would like to thank Mert İmre for his friendship, his support on anything, and for being with me in my first time of staying awake till morning for a publication. Many thanks to Ahmet Tekden and Yunus Şeker for being always available to discuss about work and their help. My genuine gratitude to my friends, Cansu Cemre Yeşilçimen, and Merve Taplı for their support and friendship since my undergraduate years. My thanks to Mert Tiftikci, Erhan Çağırıcı, and Hakan Girgin for helping me get into this new environment. Many thanks to COLORS Lab members Tuluhan, Alper, Ece, Utku, and Selçuk, with whom we created a pleasant working environment. My thanks to all my TA colleagues from BOUN CmpE, especially Binnur Görer, Hakime Öztürk, and Rıza Özçelik, for making teaching fun. I thank Manuel Del Verme for fun chats and for sharing his expertise with me.

Finally, my profound and sincere gratitude to my parents for their support, encouragement, and love. My special thanks to my sister for making everything delightful and for her immense love. You were always there for me.

This research has received funding from JST CREST project "Cognitive Mirroring" (Grant Number: JPMJCR16E2). Melisa İdil Şener received scholarship from TÜBİTAK National Scholarship Programme for MSc Students.

ABSTRACT

OBJECT, ACTION, AND OUTCOME BLENDING LATENT SPACE EXPLORATION WITH INTRINSIC MOTIVATION TO LEARN MANIPULATION SKILLS

In quest of making artificial agents more autonomous and intelligent, equipping them with the ability of self-learning of skills plays a crucial role. In this thesis, we focus on intrinsically motivated exploration to enable efficient acquisition of skills for artificial agents. During the exploration, the agent uses the intrinsic motivation signal to self-select the exploration regions to proceed. This motivation signal drives the agent to explore the region that is neither too easy nor too difficult for the agent. First, we proposed a method that continuously partitions the sensorimotor space using the predictability principle to form specialized learning regions to better employ an existing intrinsic motivation framework. Our next study aims to utilize a latent space that facilitates the self-organization of the exploratory behaviors driven by the intrinsic motivation to learn a set of skills. To make this space reflect the dynamics of the interaction between the robot and the environment, we propose blending the outcome, action, and object information. Next, the latent space is clustered into different regions; each is then learned by separate predictors. The proposed approach is validated with a simulated robot that manipulates different objects using parameterized actions in a table-top environment. Our approach allows the robot to organize its own curriculum, enabling it to proceed from easier skills to more complex ones. The analysis of the curriculum deduces that grasp emerges before pushing, which is consistent with the skill emergence in infants. Furthermore, results show that the proposed method makes significantly lesser prediction errors than its counterparts in various settings.

ÖZET

NESNE, EYLEM VE SONUÇ BİLGİSİNİ HARMANLAYAN SAKLI UZAYDA MANİPÜLASYON BECERİLERİNİN İÇSEL MOTİVASYONLU KEŞİF İLE ÖĞRENİMİ

Yapay ajanları daha özerk ve zeki yapma arayışında, onların, becerileri kendi kendilerine öğrenebilme yeteneğiyle donatılması çok önemli bir rol oynar. Bu tezde, yapay ajanların becerileri verimli bir şekilde kazanmasını sağlamak için içsel motivasyonlu keşfe odaklanmaktayız. Keşif sırasında ajan, keşfe devam edeceği bir sonraki bölgeyi içsel motivasyon sinyalini kullanarak seçer. Bu motivasyon sinyali, ajanı, ajan için ne çok kolay ne de çok zor olan bölgeyi keşfetmeye yönlendirir. İlk çalışmamızda, mevcut bir içsel motivasyon mimarisini daha iyi kullanmak amacıyla, özel öğrenme bölgelerini oluşturmak için, duyumotor uzayını öngörülebilirlik ilkesini kullanarak sürekli olarak bölen bir yöntem önerdik. Bir sonraki çalışmamız, bir dizi becerinin öğrenimi için, içsel motivasyonun yönlendirdiği keşifsel davranışların kendi kendine örgütlenmesini kolaylaştıran saklı bir uzay kullanmayı amaçlamaktadır. Bu uzayın robot ve çevre arasındaki etkileşimin dinamiklerini yansıtmasını sağlamak için, sonuç, eylem ve nesne bilgilerinin harmanlanmasını öneriyoruz. Daha sonra, bu saklı uzay farklı bölgelere ayrılmakta ve her bölge ayrı tahmin modelleri tarafından öğrenilmektedir. Önerilen yaklaşım, masa üstü bir ortamda, parametrik eylemler kullanarak farklı nesnelerle etkileşime giren, simüle edilmiş bir robotla doğrulanmaktadır. Sunduğumuz yaklaşım, robotun kendi müfredatını düzenlemesine olanak tanıyarak, robotun daha kolay olan becerilerden daha karmaşık olanlara geçmesini sağlar. Oluşan müfredatın analizi, itme becerisinden önce kavrama becerisinin ortaya çıktığı sonucuna varmaktadır, bu da bebeklerdeki beceri gelişimi ile benzerlik göstermektedir. Ayrıca, sonuçlar, önerilen yöntemin çeşitli koşullar altında, benzerlerinden önemli ölçüde daha az tahmin hatası yaptığını göstermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A	Action Vector
E	Effect Vector
E_{obs}	Observed effect vector
E_{pred}	Predicted effect vector
$e_n(t)$	Error of n^{th} region at time t
f	Sound frequency parameter
$F(\cdot, \cdot)$	F-Distribution
h	Horizontal speed parameter
I_{enc}	Image encoding
K	Number of bootstrapping samples per region
M	Motor parameters
p	p-value
PE_j	Set of potential errors when splitting j^{th} dimension
$pe_{j,i}$	Potential error by splitting j^{th} dimension from i^{th} cutting value
R_n	n^{th} region
r	Radius
r_{path}	Trajectory length
S	State vector
$S'(t+1)$	Predicted next state at time t
SM	Concatenation of state vector and motor parameters
t	Learning step
v	Vertical speed parameter
x	X position
y	Y position
z	Z position
α	Time window parameter
γ_n	Mean error of region n

Δx	X position change
Δy	Y position change
Δz	Z position change
$\overrightarrow{\Delta\phi_z}$	Z orientation change vector
θ	Smoothing parameter
κ	Batch size in the bootstrapping phase
μ	Mean value
$\mu(z)$	Mean vector of latent distribution
σ	Standard deviation value
$\sigma(z)$	Standard deviation vector of latent distribution
τ	Time window parameter
ϕ_{path}	Approach direction
$\mathbb{SM}$	Sensorimotor space
$\mathbb{R}$	Real number

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
ActionIM	Action Space Clustering Intrinsically Motivated Exploration
ANN	Artifical Neural Network
ANOVA	Analysis of Variance
API	Application Programming Interface
CB-IM	Competence-Based Intrinsic Motivation
CNN	Convolutional Neural Network
CPU	Central Processing Unit
FM	Forward Model
GAN	Generative Adversarial Networks
GMM	Gaussian Mixture Models
GPU	Graphics Processing Unit
HRL	Hierarchical Reinforcement Learning
IAC	Intelligent Adaptive Curiosity
IM	Intrinsic Motivation
IMGEP	Intrinsically Motivated Goal Exploration Processes
KB-IM	Knowledge-Based Intrinsic Motivation
K-NN	K-Nearest Neighbors
LatentIM	Latent Space Clustering Intrinsically Motivated Exploration
LP	Learning Progress
MLP	Multilayer Perceptron
MSE	Mean Square Error
PCA	Principal Component Analysis
PE-IAC	Potential Error-Based Intelligent Adaptive Curiosity
RandomIM	Randomly Clustering Intrinsically Motivated Exploration
ReLu	Rectified Linear Unit
RGB	Red Green Blue

RL	Reinforcement Learning
ROS	Robot Operating System
StateIM	State Space Clustering Intrinsically Motivated Exploration
Tukey's HSD	Tukey's Honestly Significant Difference
V-IAC	Variance-Based Intelligent Adaptive Curiosity
V-REP	Virtual Robot Experimantation Platform
VAE	Variational Autoencoder

1. INTRODUCTION

From the moment they are born, babies begin learning about their bodies and the environment autonomously. Even when there is no immediate reward or explicit assistance from their caregiver, it is quite interesting that they conduct this learning process and develop sophisticated skills. Infants learn through exploration (*online*), throughout their life (*continuous* and *open-ended*) and they acquire cognitive abilities in parallel reciprocally (*cross-modal*) [1]. Inspired by the properties of human development, this thesis studies how a robot can learn the outcomes of its actions by autonomous exploration. When there exist various interaction schemes involving different types of objects and different types of actions, the robot needs to identify different interaction schemes. This study proposes an approach for identifying such different scenarios; each is then learned by separate predictors. The autonomous exploration is driven by intrinsic motivation (IM), which enables the robot to learn these different interaction schemes by following a self-organized curriculum.

Autonomous exploration has been regarded as an essential mechanism for the learning and development of living organisms [2]. Exploratory behaviors, which enable us to adapt to different kinds of situations, learn complex skills, and practice our creativity, are observed not only in humans but also in other animals [3, 4]. Earlier, this phenomenon was studied within drive [5] and instinct [6] theories [3]. Proponents of drive theory supposed that the activities of an organism are caused by the need for preserving the homeostasis. According to Freud [7], *life instinct* makes humans engage in activities for self-preservation and survival of the species. However, White [3] states that exploratory behaviors cannot be obtained only by drives and instincts. Based on the experimental evidence, White [3] argues that these exploratory behaviors have the aim of efficient interaction with the environment. They stress that since these behaviors have a particular aim and they are persistent, they have a motivational aspect. According to Deci [8], exploration, novelty-seeking behaviors, and play stem from the human need for feeling competent and self-determining and argue that these behaviors are "intrinsically motivated".

In this thesis, we show how an agent learns purposeful skills by autonomous exploration driven by intrinsic motivation. Regarding high-dimensional search spaces, random exploration is a slow and costly process [9, 10]. However, it is shown that exploration guided by the intrinsic motivation enables an agent to focus on the problems with the appropriate level of complexity. In other words, the agent does not waste its learning time with neither too easy nor too difficult problems [11]; hence intrinsic motivation provides sample efficiency [10]. A particular intrinsic motivation signal, *learning progress* [12], considers the change of the prediction error, therefore drives the agent to explore the "unfamiliar" parts of the environment [1]. By utilizing learning progress as an IM signal, Oudeyer *et al.* [13] show that the robot self-organizes a complex developmental progression.

Considering realistic scenarios, an agent may interact with many different types of objects, using many different action possibilities under different environmental conditions throughout its life. In such a case, predicting how the environment changes based on the actions of the robot can be quite challenging. How animals and humans address this challenge inspires our study as well as several computational approaches [14, 15]. Kawato [16] investigates the existence of separate internal models in humans and animals to learn and control different objects and environments. According to Kawato [16], an internal model comprises a forward model that predicts the sensory outcome of a given action command and inverse model that estimates the action for the desired sensory outcome. Later, Wolpert and Kawato [15] argue that for motor control, humans employ multiple modules where each of them is responsible for different contexts. They indicate that this modular strategy gives rise to (1) efficient coding of the world considering all the interactions with qualitatively different contexts, (2) simultaneous learning of different contexts without interference with each other, (3) the possibility of learning a more complex context by reusing the knowledge obtained from other modules.

Exploiting modularity in learning models, now the question is: How to determine the distinctive characteristics of an environment that can be learned by separate modules? In our first work, similar to the approach proposed by Oudeyer *et al.* [13], we

identified these distinctive characteristics directly from sensorimotor space. The other studies show such an identification by learning an embedding from sensory space [17–19] and from salient events [20]. Some other studies do not consider such identification and use engineered separation [21, 22]. However, a current limit of existing approaches is that they do not take into consideration the overall picture. For example, consider "motor babbling" [23] and "goal babbling" [24] mechanisms for skill development. The former allows learning of motor and outcome associations by executing random motor commands, i.e., it explores the motor space. The latter allows learning of a motor skill by repeatedly working on fulfilling multiple goals related to that skill, i.e., it explores the outcome space [24]. Our next study aims at combining the advantages of both of the methods and including object-related information. To this end, we consider utilizing a latent space that blends outcome, action, and object-related information to assign the regions with different characteristics of the environment to disjoint forward models. It is known that, in cognitive systems, mental representations and skill learning evolve together. However, for simplification, our implementation assumes that the latent space is not procured in parallel with skill acquisition.

Our main contribution in this study is that we propose an approach for distinguishing between different characteristics of the environment so that it can exploit modularity in skill learning efficiently and effectively. To this end, our method utilizes a compact representation that blends outcome, action and object-related information. Additionally, our study (1) shows that the disjoint regions created from this latent space correspond to semantically meaningful primitives, (2) presents an active learning scheme among these primitives driven by intrinsically motivated exploration and (3) demonstrates that the developmental order arisen by this active learning scheme is in parallel with the emergence of the manipulation skills of the infants. The proposed method is validated on a simulated manipulator robot that interacts with different types of objects in a table-top environment. The results revealed that the prediction performance is better with latent space exploration than its counterparts. Furthermore, intrinsically motivated exploration in the latent space allowed us to observe that grasping emerges before pushing, which is in parallel with the motor development of the infants.

The rest of this thesis is structured as follows: in Chapter 2 the computational methods used in this study are introduced, in Chapter 3 the physical and software components used in our experiments are presented, in Chapter 4 the related studies in the literature are reviewed and in Chapter 5 our first study on this topic is presented. Chapter 6 presents the our next proposed system and explains the components of this system. Chapter 7 explains the experiment setup created for validating the proposed approach and Chapter 8 gives the experiment results. Finally, Chapter 9 provides discussions about the proposed method and Chapter 10 concludes this thesis.

2. BACKGROUND

This chapter presents an overview of computational methods and architectures used in this thesis. Each section of this chapter explains one particular method, and the subsections present a more specific version of the mentioned method.

2.1. K-Nearest Neighbors Algorithm

The K-Nearest Neighbors (K-NN) algorithm is a machine learning algorithm that considers the closest data points to calculate the output of the given input. It is a lazy learning algorithm that does not have a model-building phase, but the prediction phase usually takes a long time.

K-NN can be used for both classification and regression problems. For classification problems, the input is classified by assigning the most frequent label among its K number of neighbors. For regression problems, the weighted average of the k-nearest neighbors is used for calculating the continuous output. In this algorithm, K is a hyperparameter defined by the user, and the distance metric is a hyperparameter used for determining the neighborhood.

The advantages of this algorithm: (1) it can be applied to the data independent from its distribution, (2) it is straightforward to understand and implement. Disadvantages of this algorithm: (1) hyperparameter selection requires expertise, (2) prediction phase takes a long time as the dataset grows, (3) requires a large number of samples for better performance, (4) memory requirement is high.

2.2. Artificial Neural Networks

Artificial Neural Networks (ANN) are a collection of artificial neurons that are inspired by the biological neurons in a biological brain. The artificial neuron concept is first introduced by McCulloch and Pitts [26]. Figure 2.1 shows a schematic model of an

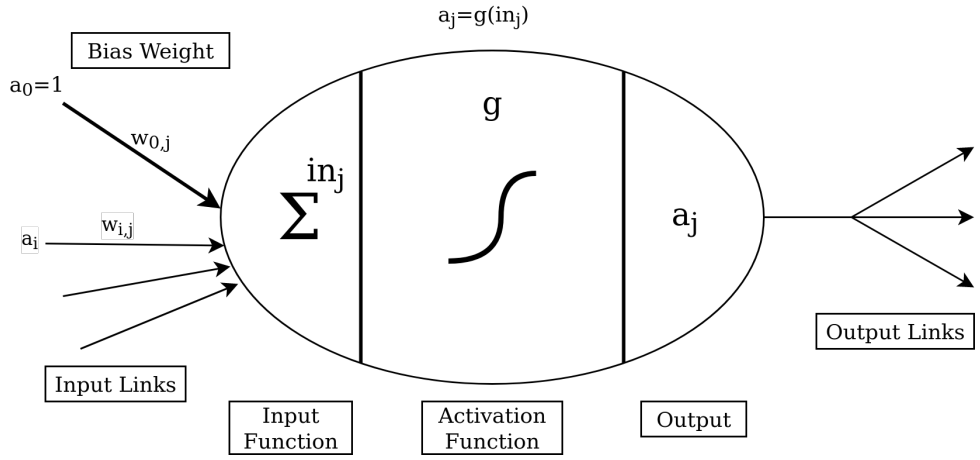


Figure 2.1. Schematic model of an artificial neuron. The figure is adapted from [25, Chapter 18].

artificial neuron. The linked structure of the artificial neural networks allows propagating the activation from an artificial neuron to another. Each connection from a neuron a_i to a_j has a weight $w_{i,j}$ that determines the contribution of the i^{th} neuron's activation to j^{th} neuron's activation. After calculating the weighted sum of the inputs, the activation function determines whether the neuron is fired or not. The activation function is an important component of artificial neurons. The neural networks are considered "Universal Function Approximators" [27]. Activation functions allow presenting the non-linear relationship between the inputs and the outputs. If the activation function is omitted, then the neuron only presents a simple linear relationship; thus, it would have less representational power. Some example activation functions are:

- Linear Unit: Outputs the weighted sum of the inputs.
- Sigmoid Unit: Uses the logistic sigmoid function. This function squashes the activation between 0 and 1.
- Hyperbolic Tangent (tanh) Unit: It is similar to sigmoid unit. Hyperbolic tangent function squashes the activation between -1 and 1.
- Rectified Linear Unit (ReLu): It is widely used in neural networks nowadays. The output of this activation function is between $[0, \text{inf})$; it gives zero output if the input is negative; otherwise, it gives the input itself.

In the literature, there are two different ways to connect the artificial neurons to create a neural network [25]:

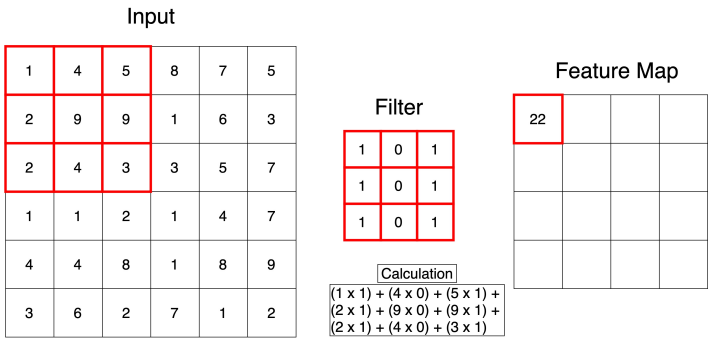
- Feed-forward Networks: They allow connections only from the input to the output. They do not exhibit loops and forms a directed acyclic graph. In order to compute the output, they only consider the input.
- Recurrent Networks: They allow feedback connections, i.e., introduce loops in the network structure. In order to compute the output, they consider the internal state of the neuron alongside the inputs.

In this thesis, the feed-forward structure of the neural networks is used. In practice, feed-forward networks are typically organized into multiple layers and are also known as Multilayer Perceptron (MLP). MLPs are widely used in deep learning applications for approximating complex functions [28, Chapter 6]. They are primarily used for capturing the patterns given large datasets and generalizing the learned non-linear relationship to the unseen data inputs.

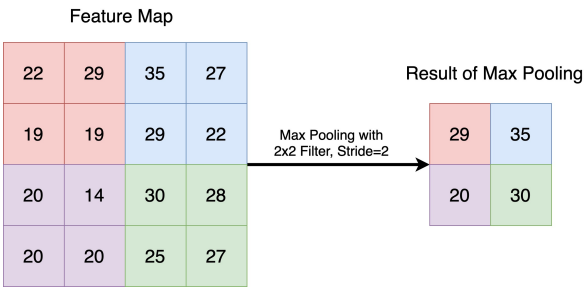
The neural networks are trained by backpropagating the error [29]. The gradient of the error is propagated from the output layer towards the input layer by altering the weights of the connections between the neurons. Besides the hyperparameters of the network architecture, hyperparameters of the training (e.g., batch size, optimization algorithm) are important for acquiring better performance.

2.2.1. Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a particular type of neural network that are commonly used for processing image, video and time-series data. They are inspired by the working mechanism of the visual cortex in the brain [30]. CNNs typically comprise convolution layers and pooling layers along with the input and the output layers. In this type of network, the convolution operation is applied in at least one of their layers [28, Chapter 9].



(a) Convolution Operation. Adapted from [31].



(b) Pooling Operation. Adapted from [32]

Figure 2.2. Schematic representation of convolution and pooling operations.

CNNs provide an efficient way of processing high-dimensional data by applying filters throughout the input. In the convolution layer, a set of filters, which have a lower dimension than the input, are passed over the data. As a filter passes over the data, it creates a feature map by calculating the weighted sum of the pixels that it passes on. A schematic overview of this operation is shown in Figure 2.2(a). In CNNs, the first few convolutional layers usually find simple features such as edges and brightness [33]. As the number of convolution layers increases, more complex features can be discovered automatically. In order to determine the activation of this operation, usually, a non-linear activation function is applied. Next, the pooling layer reduces the size of the feature map by summarizing the statistical patterns of it. An example of a pooling operation, namely Max-Pooling, is shown in Figure 2.2(b). Pooling operation provides efficiency in terms of computational resources and prevents overfitting. CNNs enable parameter sharing. Since a filter can be used anywhere in the data, it abandons the need for learning a separate set of parameters for each location. Due to pooling operation, CNNs provide translation-invariance; however, invariance to scaling or rotation can be handled by different kinds of mechanisms [28, Chapter 9].

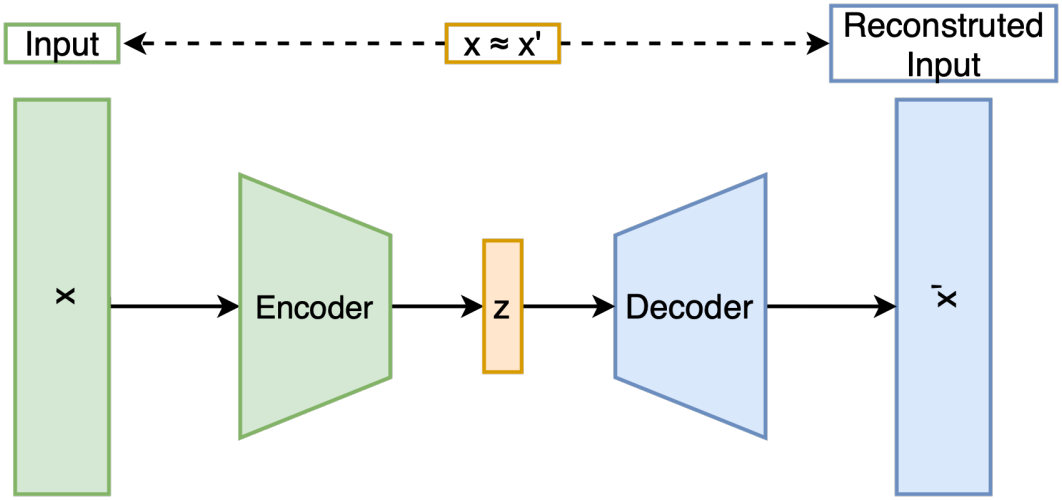


Figure 2.3. General structure of the autoencoders. The encoder encodes the input to the internal representation (z). The decoder takes the internal representation (z) and outputs the reconstruction of the input. The figure is adapted from [34].

2.2.2. Autoencoders

Autoencoders are another particular kind of neural network that is trained to reconstruct the given input. They are typically used for encoding the input to a lower-dimensional representation; hence they provide dimensionality reduction. Autoencoders are preferred to the conventional dimensionality reduction techniques (e.g., Principal Component Analysis (PCA)); because they can find complex non-linear relationships. In many cases, obtaining lower-dimensional representations are desirable; since it provides performance improvement, and it is efficient in terms of computational resources.

Autoencoders consist of two parts, as shown in Figure 2.3. The encoder part encodes the given input to the internal representation. The decoder part reconstructs the input of the encoder from the internal representation. Although the internal representation typically has a lower-dimension than the input, larger internal representations are shown to be effective in sparse coding settings [28, Chapter 14]. The parameters of the encoder and the decoder are learned jointly by minimizing the reconstruction loss, i.e., the dissimilarity between the input and the reconstructed version of the input. Thus, autoencoders are considered as an unsupervised learning technique.

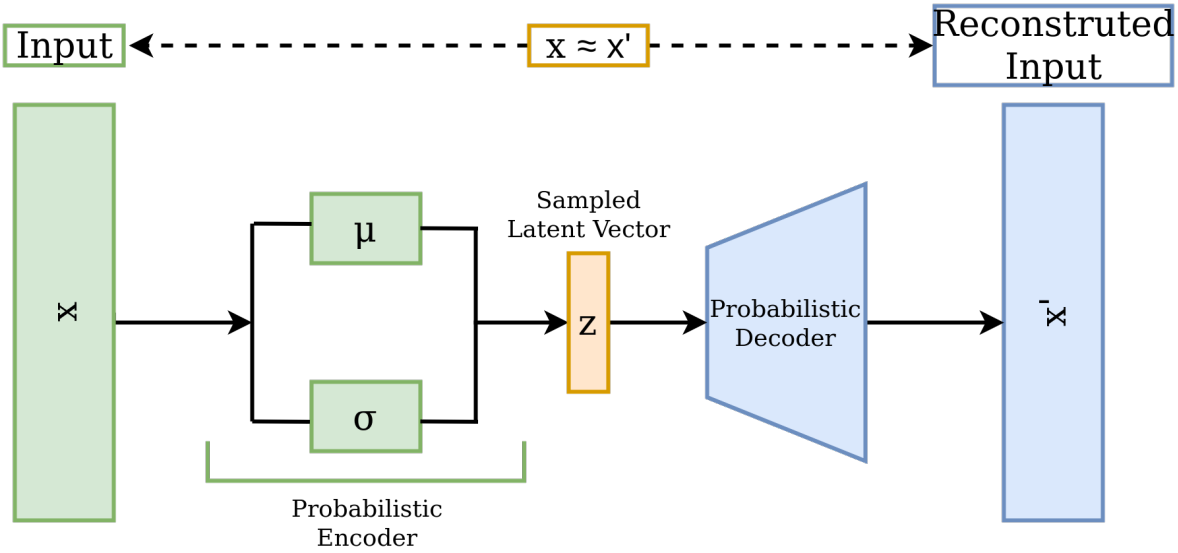


Figure 2.4. General structure of the VAE. The probabilistic encoder approximates $q_\phi(z|x)$, and the probabilistic decoder approximates $p_\theta(x|z)$. The figure is adapted from [34].

2.2.2.1. Variational Autoencoders. Variational Autoencoders (VAE) [35] are a type of autoencoders which are based on variational inference. The encoder part of a VAE maps the given input into distribution in the latent space. The decoder part decodes a point sampled from the encoded distribution. A general structure of VAEs is shown in Figure 2.4. If the distribution is assummed to be parameterized by θ, the likelihood of generating real data samples ($\mathbf{x}$) from the latent encoding vector ($\mathbf{z}$) would be $p_\theta(\mathbf{x}|\mathbf{z})$. Similarly, the prior and posterior would be $p_\theta(\mathbf{z})$ and $p_\theta(\mathbf{z}|\mathbf{x})$. Since the integral of the marginal likelihood is intractable, Kingma and Welling [35] proposed using an approximation $q_\phi(\mathbf{z}|\mathbf{x})$ parameterized by ϕ instead of intractable posterior $p_\theta(\mathbf{z}|\mathbf{x})$. VAEs are trained by both maximizing the log-likelihood of generating real data $p_\theta(\mathbf{x})$ and minimizing the difference between the approximate and real the posterior distribution $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$ [34], as shown in Equation 2.1. In order to use backpropagation during the training phase, *reparameterization trick* [35] is used.

$$\begin{aligned} L_{VAE}(\theta,\phi) &= -\log p_\theta(\mathbf{x}) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) \\ &= -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) \end{aligned} \tag{2.1}$$

2.3. Gaussian Mixture Models

Gaussian Mixture Models (GMM) is a probabilistic method that represents a probability distribution with multiple Gaussian distributions. Each Gaussian distribution is called as a component of the GMM. GMMs are parameterized by the weight of the mixture components, component means, and component variances/covariances.

GMMs are trained by the Expectation-maximization algorithm. In the expectation step, for each data point, the expectation of being a member of a component is calculated. In the maximization step, the parameters of the model are updated to maximize the probability of the model generating these parameters. An essential property of the expectation-maximization algorithm is that at each iteration, the maximum likelihood of the data increases. Hence it is guaranteed to find a local maximum or a saddle point. These steps repeat until the algorithm converges.

GMMs are commonly used for clustering and density estimation. GMMs are able to generate clusters with different shapes and sizes. It makes the soft-assignment of clusters. That means, for a data point, GMMs give membership probabilities of each Gaussian.

3. EXPERIMENT PLATFORM

This chapter presents the tools used in this thesis. Section 3.1 introduces the physical entities involved in this study. Section 3.2 provides the computational tools employed in this study.

3.1. Physical Units

3.1.1. UR10 Robot

In this thesis, the experiments of the proposed method are conducted on a simulated UR10 robot. Although we have used the simulated version, it is important to mention the general features of the real robot. UR10 is a collaborative industrial robot that is manufactured by Danish company Universal Robots [36]. It is capable of carrying up to 10 kilograms. The robot has six-degrees-of-freedom, and it can reach up to 1.3 meters. It is mainly used in production lines for assembly and packaging tasks.

3.1.2. Kinect

Kinect [37] is a product of Microsoft Corporation. It has an RGB color video camera and an infrared camera to determine the depth information. Furthermore, it can perform motion capturing. It is widely used in robotics applications. In this thesis, Kinect is used in simulation experiments of the proposed method to capture the depth images of the objects.

3.2. Software Units

3.2.1. Robot Operating System

Robot Operating System (ROS) [38] is an open-source software that provides various types of services to the robotic software processes. Although it is not an operating

system, similar to the operating systems, it provides hardware abstraction and allows message-passing across processes. Furthermore, it serves low-level device control, implementation of commonly-used functionality, and package management functionalities. ROS supports a large number of commonly used robots, sensors, and motors.

The runtime processes are represented in a graph structure where each process is called a node of this graph. Other entities of this graph structure are the master node, messages, services, topics, and bags. Each node can communicate with another node by messages which are passed over a topic. The services are a more specialized way of passing messages that are based on request/reply strategy. Bags are used for storing and playing back the message data. The master node establishes the connection between the graph entities and provides parameters to the nodes at runtime. A ROS system can work using multiple computers. In this thesis, ROS is used for scheduling the data collection in simulation.

3.2.2. CoppeliaSim

CoppeliaSim (formerly V-REP) is a general-purpose robotic simulator that allows distributed control of the simulated environment [39]. It provides a broad range of robots used in industry and academy, as well as widely used sensors. It also presents building blocks, e.g., actuators, grippers, sensors, for designing custom robots. Moreover, there are various types of objects and environment properties that allow simulation of real-world scenarios. The entities of the simulation scenes can be independently controlled by embedded scripts, by ROS nodes, by remote API clients or plugins. The embedded scripts are written in Lua, and ROS nodes are usually written in C/C++ or Python. CoppeliaSim has builtin calculation modules that enable various features such as collision detection, inverse kinematics calculation, path/motion planning. It provides the Bullet Physics library, Open Dynamics Engine (ODE), Newton Dynamics engine, and Vortex Studio engine for simulating physical properties of the environment. In this thesis, the experiments with UR10 robot are implemented and conducted in CoppeliaSim. For the inverse kinematics calculation, inverse kinematic module of the CoppeliaSim is used. We have used Bullet 2.78 physics engine.

3.2.3. External Library and Packages

3.2.3.1. Scikit-learn. Scikit-learn is a Python library that provides a broad range of machine learning algorithms as well as utility functions required for machine learning applications [40]. It is built on SciPy, a Python library for scientific computing, and it uses NumPy for mathematical functionalities. In this thesis, K-NN regressors and GMMs are implemented by using Scikit-learn. In this study, model selection and preprocessing utilities are also used from Scikit-learn library.

3.2.3.2. Keras. Keras is an open-source library for designing and working with neural networks [41]. It provides a broad range of implementations required for creating neural network models. It presents commonly used losses, optimizers, and metrics as well as various utilities for neural network experiments. Keras is written in Python, and it can utilize Tensorflow, Theano, and Microsoft Cognitive Toolkit (CNTK) as backend. It allows training both on the central processing unit (CPU) and on the graphics processing unit (GPU). It is widely used in both industry and the research community. The networks used in this thesis are implemented using Keras API.

3.2.3.3. Statistical Analysis Packages. In this thesis, for statistical analysis of the results several Python packages are used:

- Pingouin [42] is a Python package that provides several statistical analysis tools. It provides analysis of variance (ANOVA), several post-hoc tests and some other widely used statistical methods.
- Scipy.stats [43] is a module of SciPy library. It provides a wide array of statistical functions and probability distributions.
- Scikit-posthocs [44] is a Python package that provides a large number of post hoc tests to conduct pairwise comparisons after performing ANOVA.

4. RELATED WORK

4.1. Intrinsic Motivation

Intrinsic motivation causes an agent to seek novel, surprising phenomena, and make it focus on developing the skills with intermediate complexity [4]. As it fosters exploratory behavior in the organisms, it is considered that intrinsic motivation is an essential element of open-ended learning [2]. The ability to seek information by exploration, driven by the pursuit of novelty and surprisingness observed in infants and other animals [3, 4].

Inspired by the intrinsic motivation in psychology [45], researchers developed computational models of intrinsic motivation to equip artificial agents with the ability of open-ended and autonomous learning [12, 13, 46, 47]. Oudeyer and Kaplan [2] divide the computational approaches of intrinsic motivation (IM) into two classes as *Knowledge-Based IM (KB-IM)* and *Competence-Based IM (CB-IM)*. KB-IM strategy is derived from the deviation of the knowledge of the agent on the environment from reality [2]. While the agent expands its knowledge about the environment by exploring the situations outside of its current understanding, it learns new skills [48]. The CB-IM strategy focuses on a specific state, i.e., the goal state, that is adaptively changed according to the current competencies of the agent [48]. In other words, the CB-IM stems from the performance of an agent to achieve a specific goal. Mirolli and Baldassarre [49] state that both of KB-IM and CB-IM can serve knowledge and competence acquisition. They also point out that they might have a complementary relationship in a sense that the KB-IM mechanisms to detect which skills to train based on their novelties and the CB-IM mechanisms to select the expert to achieve a particular goal.

4.1.1. Computational Models of Intrinsic Motivation

Blank *et al.* [46] stress that the "inherent anthropomorphic bias" is a problem of task-oriented robot design methodologies. They state that expecting a robot to

accomplish a task as humans do is wrong, and they propose an intrinsic developmental algorithm to allow the robot to discover its capabilities on its own. The robot first builds abstractions from raw sensory data, which enable it to focus on the most crucial features of the environment. Then it utilizes the abstractions to predict the future. The motivation mechanism enables the robot to learn complex behaviors from low-level control schemes. Similar to their work, our proposed approach forms abstractions later to be used in the motivation framework.

Oudeyer *et al.* [13] propose a mechanism that is called "Intelligent Adaptive Curiosity" that drives the agent to concentrate on situations with intermediate complexity to foster autonomous mental development. Throughout the exploration, the agent masters the environment dynamics and partitions the sensorimotor space to be further learned by the local experts. To build a curriculum, the agent makes use of an intrinsic reward measure, particularly the learning progress, to select which region to explore next. Thus, the agent distinguishes the regions that are controllable from uncontrollable ones and develops a curriculum with an increasing level of complexity within the controllable part. This study is one of the pioneering work in intrinsically motivated learning in robotics and provided an example of computational architecture and its implementation for the next generations of open-ended learning in artificial systems [50, 51]. Our first study is inspired by this study in terms of partitioning the exploration space into the regions and using disjoint forward models that are exclusive for one particular region. Similar to their work, we consider learning progress as the intrinsic motivation signal.

In the proposed approach, there is no predefined task, and the main interest is to make effect-prediction using forward models. However, in the literature, there exist reinforcement learning studies that utilize intrinsic motivation to learn a policy to accomplish a given task. Chentanez *et al.* [52] propose using the intrinsic motivation in the Reinforcement Learning (RL) domain to create more competent agents that are able to develop and enhance the hierarchies of skill learning, as stated by Barto *et al.* [53]. Uchibe and Doya [54] use embodied evolution of a group of mobile robots to discover intrinsic motivation in an RL framework. Hester and Stone [55] use a combi-

nation of intrinsic motivation signals in an RL framework to discover novel states and improve the performance of the model in the states where it is uncertain. Florensa *et al.* [56] suggest a self-supervised RL framework that generates goals with intermediate complexity using Generative Adversarial Networks (GAN) to produce a curriculum for the agent automatically. To learn complex skills with minimal supervision, Eysenbach *et al.* [57] express that the agent should learn skills as diverse as possible. To achieve that, they formulate "diversity" with an information-theoretic objective to guide the exploration of an unsupervised RL agent. Colas *et al.* [58] propose an intrinsically motivated RL architecture that allows the robot to achieve multiple goals with increasing complexity using a single policy. To address the problems of large action and state spaces in RL, Hierarchical Reinforcement Learning (HRL) [59] use abstractions to simplify the problem. Intrinsic motivation is also applied to HRL studies at the different levels of the hierarchy [20, 60–63], both in high-level and low-level controllers.

4.1.2. Computational Models of Infant Development

The studies mentioned so far are not involved much in observing a developmental order and relating their findings with psychological research. Several works put emphasis on infant development in computational models of intrinsic motivation [13]. Forestier *et al.* [21] develop an algorithmic procedure called "intrinsically motivated goal exploration processes" (IMGEP) that allows the autonomous discovery and selection of goals. These goals are achieved by following a self-generated curriculum with an increasing level of complexity. Their method uses reward functions as given modular representations of goal spaces and explores them by employing intrinsic motivation. They validate their approach in a 3D printed robot that first learns how to move its end-effector and then use tools to learn more complex relationships that are present in the experiment setup. Haber *et al.* [64] show a computational model where the agent develops the understanding of ego-motion, followed by the ability to interact with single and multi-object. To discover the most informative parts of the environment, they utilize a "world model" that is trained with a history of states and predicts the action. They use a model that proposes actions through a "loss model" which predicts the

loss of the world model to drive the exploration to discover novel states. Mannella *et al.* [65] hypothesize that an agent learns about the dynamics of its body by autonomous goal generation regulated by the intrinsic motivations. To validate their hypothesis, they create a model that relies on CB-IM signal to form abstract representations of the observations, to select goals to pursue, and to learn motor skills.

Nagai [66] suggests that predictive learning of sensorimotor information allows infants to perform goal-directed actions and to obtain some other cognitive abilities and propose a computational architecture for predictive learning. In their study, the forward model predicts the next sensory state and action to be executed, given the current sensory state and the action. They show that predictive learning enabled the hierarchical development of goal-directed actions. They also demonstrate that as the performance of the predictor improves, the robot develops grasping and reaching abilities. Gaussier and Pitti [67] shows a neurocomputational mechanism for multi-modal integration in infants to reach and grasp objects. They hypothesize that "reachable regions" cells, which are considered to be organized in the brain by sensorimotor contingencies, manages the learning of reaching and grasping skills.

Bugur *et al.* [68] show the emergence of push and grasp behavior on a simulated robot in a 3D environment is parallel to the infant motor development. They use the action and effect space information to obtain a latent representation that eventually distinguishes between two high-level actions, namely push and grasp. The latent representation is formed by first, obtaining two clusters from the effect space, and second, by using the cluster information of the effect space, applying Linear Discriminant Analysis on the action space. Throughout the exploration, they further partition this latent space, which results in multiple local experts. Similar to their study, we learn local forward models to capture the dynamics of regions of the latent space and exploration is driven by the learning progress. However, our study differs from their study by means of forming the exploration regions. In our study, we are integrating three different modalities, namely object, action, and outcome in the latent space, and we do not further partition this latent space during the exploration.

4.1.3. Robot Learning Studies Exploiting Intrinsic Motivation

Ivaldi *et al.* [69] introduce an active perception system that utilizes curiosity-driven exploration for manipulation tasks. Their method uses socially guided learning along with the curiosity-driven exploration decide on which object to interact with which manipulation strategy to recognize the objects. Oudeyer *et al.* [70] present a method that combines intrinsic motivation and imitation learning for motor skill development in robots. They argue that social guidance can allow an intrinsically motivated learner to learn areas that are difficult to discover faster. Combining socially guided learning and intrinsic motivation is also used in hierarchical learning of interrelated tasks [71]. Fournier *et al.* [10] introduce an RL agent that learns multiple tasks by learning from demonstrations of another agent and actively chooses what task to explore in a non-rewarding environment.

Ugur *et al.* [72] propose a curiosity-driven learning scheme to speed up the learning of the traversability affordance for a mobile robot. Ugur and Piater [73] suggest a system to structure and learn interdependent affordances in a table-top manipulation scenario by intrinsic motivation. They used intrinsic motivation for action selection, object selection, and feature selection to learn affordances. Baldassarre *et al.* [74] introduce an architecture to learn affordances that are to be used in solving planning tasks. They propose learning affordances through intrinsic motivation and using attention-based active perception to decompose the planning tasks to learned affordances. Blaes *et al.* [22] utilize intrinsic motivation in a task-planning architecture. They assume that the environment is partitioned into controllable goal spaces, and the agent selects between these spaces using surprise-based intrinsic motivation.

4.2. Representation Learning

One particular aspect of our study is to learn a representation that makes the exploration more efficient by simplifying the task into easier parts, which are then explored by intrinsic motivation. In a comprehensive review of representation learning [75] it is defined as "learning representations of the data that make it easier to ex-

tract useful information when building classifiers or other predictors". Most of the work in robot learning practices engineered features representations to perform given tasks. However, to obtain full autonomy in intelligent systems, the agent also should be capable of building efficient feature representations from raw sensory data. Representation learning in robotics is an important research direction that allows the learning systems to be efficient in computational resources, generalization ability, time efficiency, and abandons the need for feature engineering. Many works from robot learning, control, and RL domains focus on learning such representations to foster autonomy. Bowling *et al.* [76] present an example of state representation learning that utilizes the action information to the trajectory prediction task. Boots *et al.* [77] proposes learning "predictive state representation" by making use of action and observation trajectories later to be used in a robotic planning task. Lange *et al.* [78] investigate whether using an autoencoder to obtain state representation from image data is useful for learning a control policy. [79, 80] suggest using "robotics priors" to learn an appropriate state representation to facilitate the learning of environment dynamics. Watter *et al.* [81] and [82] suggest embedding high-dimensional observations to be used in complex control tasks. Machado *et al.* [83] propose an algorithm for discovering options [84] by using successor representation [85]. Whitney *et al.* [86] suggest learning an abstraction that groups state and action pairs with similar outcomes to attain sample efficiency in RL frameworks.

Several works consider using representation learning approaches along with the intrinsic motivation. Mohamed and Rezende [17] propose a formulation for estimating mutual information, which is used as an IM signal, namely the "empowerment", in an RL framework. They aim to develop a vision-based motivation system and apply representation learning to encode raw sensory observation into the state. Santucci *et al.* [20] introduce a hierarchical architecture that allows the agent autonomously to discover goal states and to choose which goal to pursue by using a CB-IM signal in a 3D simulated setup that involves a robot learning to reach the objects. They build an implicit and an explicit representation from the changes observed in the environment to make an RL agent pursue them as candidate goals. Vezhnevets *et al.* [61] suggest a two-level architecture in which the higher-level controller sets goals for a

lower-level controller, which is intrinsically motivated to fulfill these goals. They use a latent state space to identify the sub-goals. Péré *et al.* [19] added a representation learning algorithm on top of IMGEP [21] to create the goal-spaces from raw sensory observations. In that study, the agent passively observes the environment to collect data about the environment then uses a representation learning algorithm to learn an embedding function. After that phase, learned representation is used as the goal space to be explored by the intrinsically motivated architecture they proposed previously. Later, Laversanne-Finot *et al.* [18] use a disentangled representation learning stage to form goal spaces to be used in IMGEP [21]. They observe that the agent could discover and learn the controllable goal spaces in the presence of the distractor sensory phenomena. To provide exploration efficiency, Blau *et al.* [87] introduce Bayesian curiosity to the RL domain. Their method brings the observations to a latent space that is being used for computing the intrinsic reward (i.e., uncertainty) of the observation. Then, the policy network is trained using a combination of extrinsic and intrinsic reward signals. Hafez *et al.* [88] propose a system that learns a latent space that is inspired by the human mental simulation of motor behavior. They use this latent space to supply "imagined experiences" as training data to local dynamics models to improve their predictions. They determine the imagined experiences by utilizing intrinsic motivation. Zhao *et al.* [89] use embeddings of actions and observations to find a linear dynamics model in action embedding space. They formulate that the learned embedding captures the information-theoretic properties of the environment, and use this quantity as the intrinsic reward signal to augment the reward for the RL algorithms.

5. UTILIZING PREDICTABILITY PRINCIPLE FOR INTRINSICALLY MOTIVATED EXPLORATION OF THE SENSORIMOTOR SPACE

This chapter presents our first study on intrinsically motivated exploration strategies. Section 5.1 presents the context of our first study, Section 5.2 provides the details of the method, Section 5.3 explains the experiment setup and Section 5.4 gives the results of the experiments.

5.1. Introduction

For many years, scientists generally have followed three main approaches for building intelligent systems [90]. In the first one, an intelligent system is directly programmed to perform a given task. In the second, the computer is provided human-edited sensory data and runs a learning program specific to the task. Finally, in the last, intelligent systems evolved by the principle of "survival of the fittest", i.e., the most competent races left their survival skills to successive generations [90].

Survival depends on lifelong learning and application of what has been learned. Intrinsic motivation (IM) is regarded as a set of active learning mechanisms for developmental robots, improving learning in high-dimensional search spaces. Since IM demands the development of broad competence rather than immediate external goals [53], IM is a part of continuous and high-quality learning [91]. The autonomous mental development concept is defined as developing mental capabilities under the control of a learning agent's own developmental program, via the autonomous real-time interactions with the external environment with the sensors and actuators of the agent as well as its internal environment with time [90]. Originated from the fact that IM mechanisms generate learning signals by observing the skills or knowledge level needed to be acquired by the agent [92], autonomous mental development concepts were tried to be adapted to learning machines.

Weng *et al.* [90] state that, contrary to manual development involving running a program for a specific task with hand-engineered representations, autonomous development consists of two main phases. According to Weng *et al.* [90], the first one (construction and programming phase), a developmental program is formed, controlling the autonomous development of the agent and not related to a specific task, and the agent's body is designed according to its operating environment. The beginning time of the execution of this program is considered the time that the agent was "born" and the second phase begins. In the second one (autonomous development phase), the agent starts interacting with the physical world and develops the skills required in that environment. In this scheme, Weng *et al.* [90] argues that the skills learned in the earlier stage of the agent's lifetime form a basis for learning new skills.

Efficient and effective learning in high-dimensional spaces is hard and can be simplified by splitting the sensorimotor space ($\mathbb{SM}$) into smaller regions. The regions with similar characteristics can be generalized effectively by an expert responsible only for these regions. Distributing learning tasks across the experts of smaller regions provides more accurate results and thus improves the overall quality of the learning. In order to make these experts proficient in their local regions, $\mathbb{SM}$ should be split wisely. A particular method of IM, namely Intelligent Adaptive Curiosity (IAC), provides a smart splitting scheme in such a high-dimensional $\mathbb{SM}$ and drives the learning agent to explore the regions by considering the competence level of the agent. The essential point that forms the core of our study is the decision of how to split $\mathbb{SM}$ into specialized learning regions. Reflecting environment dynamics to the learning space and formation of child regions should be determined by the previously collected experience of to be split region. Thus, to determine the regions to be formed after the split, we are considering predictability, which is the degree of potential success rates of the candidate regions. Our study performed better than the original study [13] (we will also briefly explain their idea in this chapter) does not consider how the experts would perform in the generated sub-regions, in an experimental setting simulating the interaction of a robot with a simple 2D environment. In this study, we propose a novel method to split the learning space into easy to learn regions and provide a more accurate way of calculating the intrinsic reward that the agent gets. As a result, our approach considers

the future aspects of the splitting process and reflects a more distilled way of using IM for exploration.

5.2. Method

The proposed method is built on the IAC framework proposed by Oudeyer *et al.* [13]. In Subsection 5.2.1 we give a brief introduction to IAC, in Subsection 5.2.2 we explain the method given in [13] and explain our approach of splitting $\mathbb{SM}$, in Subsection 5.2.3 we describe learning machines and their contribution to the splitting process, then in Subsection 5.2.4 we provide the formal explanation of the learning progress and finally in Subsection 5.2.5 we explain the action selection mechanism.

5.2.1. Summary

Intelligent adaptive curiosity (IAC) proposed by [13], adaptively splits $\mathbb{SM}$ into regions and uses LP of these regions for deciding which region to explore and learn next. Regions with large LP are primarily explored and learned. Since LP would be low in problems that are too easy or too complex or impossible to learn, it automatically works on problems that have moderate complexity before dealing with simple and hard learning problems. Thus, the actions of the robot become more complex gradually, and the developmental sequence organizes itself.

The flow of the IAC algorithm can be summarized as follows: Each experience encountered by the robot is recorded to the memory of the system as a vector (we will call them as "exemplar" in order to be coherent with IAC [13]). An exemplar is a couple of current sensorimotor state and its outcome in sensory space $\langle SM(t), S(t+1)\rangle$. $\mathbb{SM}$ continuously split into regions when any region met splitting criterion. Note that splitting can be any condition depending on the application, here we used a threshold value for the number of exemplars that a region is allowed to contain as in [13]. Each region has its learning machine, and this machine is responsible for predicting the next sensory state $S(t+1)$ given the current sensorimotor state $SM(t)$ covered by that region. Each learning machine is trained with the exemplars of the corresponding

region, and when a prediction should be made, the learning machine covering that exemplar is selected and used for the prediction. After the execution of the action in the given sensorimotor context, the difference between the actual outcome and the prediction is calculated and recorded into the error list of the corresponding region. Afterward, this list is used for the evaluation of the LP of that region. LP is the core of the IM in the system and used for the determination of the action, which contributes most to the learning process.

5.2.2. Splitting the Sensorimotor Space

We aimed to improve the learning performance of the IAC by splitting $\mathbb{SM}$ according to the predictability principle. IAC splits $\mathbb{SM}$, and the mechanism behind this division is an essential part of our study. In our method, before the actual split performed, the regions are hypothetically split into two parts a predefined number of times by considering each $SM(t)$ dimension (feature) and potential learning success of each hypothetical region is calculated. This process clarifies our idea of determining which feature dimension and value will be used in that region's splitting procedure.

The splitting procedure can be summarized as follows: In the beginning, only one region (R_0) exists, and when it meets the splitting condition, it is split into two new regions. This way, each region, when it satisfies the splitting condition, is split into two child regions and stores the feature dimension used for splitting along with the corresponding cutting value in itself. Since the cutting dimension of a region corresponds to a feature of SM vectors, when a prediction is to be made, the corresponding region can easily be found by using the cutting dimension and value stored in regions. After splitting, exemplars contained by a parent region distributed across its children by considering the cutting information. For example: if the selected cutting dimension is the motor command and the determined cutting value is 0.5, all the exemplars inside the left child region would have their motor command value below 0.5 while all the exemplars inside the right region would have their motor command value above 0.5. In the rest of this chapter, we will refer our method based on the potential error calculation as PE-IAC and the method proposed in [13] based on variance as V-IAC.

In V-IAC, a region is partitioned into two new regions in which the sum of the variances of $S(t+1)$ components weighted by the cardinality of each region is minimal. A detailed explanation of it can be found in [13]. Our proposed method PE-IAC first hypothetically splits exemplar set into two by each feature dimension of SM vector predefined number (chosen arbitrarily) of times. From the hypothetical pair of regions, the pair with the lowest total potential error is selected. Thereby, instead of V-IAC, which does the splitting according to feature distribution in exemplars, splitting in PE-IAC is done by taking potential successes of candidate regions into account. Let each exemplar $SM(t)$ is a vector with length l and the decision of how to split the region is made by the following steps:

- Exemplar set of each parent region is sorted for each dimension index j by considering only that dimension.
- The sorted set of exemplars are split into two from different cutting values. Each hypothetical child region's learning machine is trained with the set of exemplars contained by that region, and corresponding errors are calculated. This process is executed incrementally. The sum of the errors of each child region is divided into the length of the exemplar set, and the minimum of these two values is taken and stored as $pe_{j,i}$ (potential error by splitting j^{th} dimension from its i^{th} cutting value). From all the calculated error rates for that dimension $PE_j = \{pe_{j,1}, pe_{j,2} \dots pe_{j,i}\}$, the smallest value is selected $pe_j = \min(PE_j)$ and corresponding cut value is stored.
- Smallest potential error rate from all dimensions is selected, and corresponding cutting dimension and cutting value are used for actual splitting.

5.2.3. Learning Machines

Each region has a learning machine that is trained by that region's exemplars. The learning machine of a region is responsible for the prediction of the next sensory state given sensorimotor input covered by the region. Any machine learning algorithm can be used for implementing a learning machine. For the sake of the integrity of the

system, the same algorithm could be used for all learning machine inside it. In this study, the selection of the learning algorithm does not depend on the method, and any algorithm that is compatible with the given learning task could be used. In [13], K-Nearest Neighbor (K-NN) [93] is used as the regression algorithm. When a region is split, learning machines of the new regions cannot use the learning machine of their parent directly. Thus, after each split, newly generated child regions should train their own learning machine with their own set of exemplars.

The second main contribution of our proposed method is that each new learning machine is trained by the exemplars of the corresponding region one-by-one and forms its own error list. Therefore, different from V-IAC, where each child inherits exemplars of its parent, in our method, each child region and its corresponding expert considers only the errors made only by itself.

5.2.4. Calculating Learning Progress

The LP of each region is computed by the approach suggested by Oudeyer *et al.* [13]. Let $S'(t+1)$ denote the prediction, $S(t+1)$ denote the actual outcome of $SM(t)$ vector and $e_n(t+1)$ denote the error. Then the error is mathematically:

$$e_n(t+1) = ||S(t+1) - S'(t+1)||^2 \tag{5.1}$$

Region R_n's error list will be consist of:

$$e_n(t+1-\phi), e_n(t+1-\phi+\omega_1), e_n(t+1-\phi+\omega_2), \ldots, e_n(t+1)$$

Here $e_n(t+1-\phi)$ denotes the error of first exemplar covered by that region and inherited from the parent. Since the exemplars inherited from the parent does not follow a regular time pattern, $e_n(t+1-\phi+\omega_1)$ denotes the next exemplar covered by that region and $e_n(t+1)$ denotes the most recent prediction error.

LP of a region is calculated by taking the smoothed derivative of the closest error curve and smoothed derivative of the older closest error curve. Let θ denote the smoothing parameter and τ denote the time window parameter, mathematically:

$$\langle e_n(t+1)\rangle = \frac{\sum_{i=0}^{\theta} e_n(t+1-i)}{\theta} \tag{5.2}$$

$$\langle e_n(t+1-\tau)\rangle = \frac{\sum_{i=0}^{\theta} e_n(t+1-\tau-i)}{\theta} \tag{5.3}$$

$\langle e_n(t+1)\rangle$ and $\langle e_n(t+1-\tau)\rangle$ denote the smoothed derivative of the closest errors and the older closest errors respectively. The LP is calculated by taking the negative of the actual decrease in the prediction error:

$$L(t+1) = \langle e_n(t+1-\tau)\rangle - \langle e_n(t+1)\rangle \tag{5.4}$$

5.2.5. Action Selection

In IM systems, action selection is made by maximizing the intrinsic reward that the agent gains from executing the corresponding action. In our problem, since $\mathbb{SM}$ is continuous, the next candidate $SM(t+1)$ vector is selected by random sampling inside this space. In a set consisting of 100 sampled exemplars, each sample's corresponding region is found, and LP of these regions are compared. With ϵ-greedy action selection mechanism, the sample covered by the region with the largest LP is selected and used as the next sensorimotor input of the system. Next, the input is executed, results are observed, and the system is updated. PE-IAC and V-IAC use the same mechanism to calculate LP and to select the action.

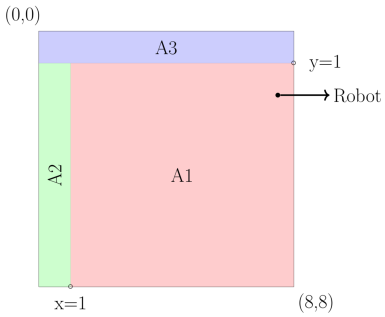


Figure 5.1. Experiment environment is a 8×8 2D environment consists of areas with different characteristics. Area-1 (A1) is slippery, Area-2 (A2) is sticky and Area-3 (A3) is completely random. The robot changes its position on the given environment.

5.3. Experiment Setup

In the experiment setting, a robot that moves in an 8×8 2D environment is simulated, as shown in Figure 5.1. The experiment environment consists of three sub-areas with each of them has a different characteristic. The consequences of the robot's actions depend on the area it stands and the frequency of the sound emitted by the robot. The robot moves in vertical and horizontal directions and motor commands of this movement, namely h, v defined in $h, v \in \mathbb{R} \mid -1 < h, v < 1$. Without considering the effect of the area that the robot stands and the frequency of the sound, if the robot's horizontal motor command $h = 0.5$ and its x position at time t is $x = 1.2$ then at time $t+1$ the robot would be in $x = 1.7$ by executing the given action. Furthermore, it emits a sound with frequency $f \in \mathbb{R} \mid 0 \leq f < 1$, and the frequency affects the interaction of the robot with the environment. Without considering the effect of the area that the robot stands, if the emitted sound frequency value is $f1 \in [0, 0.33)$, reverse of the motor commands are executed (i.e., $\langle h = 0.1, v = 0.4 \rangle \mapsto \langle h = -0.1, v = -0.4 \rangle$. If it is $f2 \in [0.33, 0.66)$ regardless of their value, executed as $h = 0, v = 0$. If $f3 \in [0.66, 1)$ then the commands executed as they are.

After considering the effect of sound frequency on robot's interactions, h, v commands executed depending on the area it stands, i.e., for A1, multiplying both with 3; for A2 dividing both with 2 and for A3 movement of the robot will be completely random. Changes in the x, y position of the robot are calculated by adding h, v com-

mands to current x, y position of the robot. In this setting $M(t) = (h, v, f)$ consists of horizontal speed, vertical speed and sound frequency respectively. Sensory vector consists of x, y position of the robot: $S(t) = (x, y)$. In brief, robot maps the next sensory state to the current sensorimotor input: $SM(t) = (h, v, f, x, y) \mapsto (x', y') = S(t + 1)$

5.3.1. Learning Flow

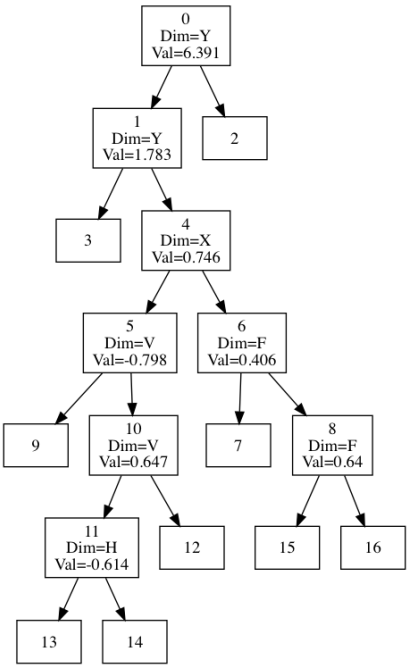
At time t, N possible SM vector is introduced by the system. Except for the time before the first split and ϵ-greedy action selection rule, the next action is determined by IM, namely, LP of the regions. (N depends on the environment dynamics, and here we used $N = 100$.) Next, the robot predicts $S(t + 1)$ with the given SM vector. In order to make a prediction, the system finds the responsible region of that exemplar, and the learning machine of that region is used for the prediction of the execution of this vector's outcome. Finally, this SM vector is stored in exemplar set of that region.

5.3.2. Experiment Parameters

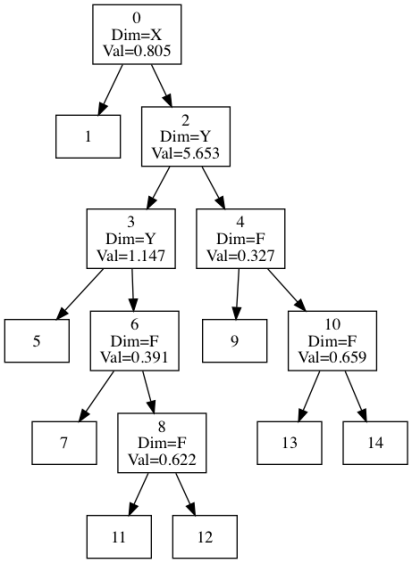
The system is trained with 5000 iterations, and the required number of exemplars for the splitting condition is 1000 (splitting criterion), i.e., when a region collects 1000 exemplars, it is split into two new regions. For ϵ-greedy action selection rule, $\epsilon = 0.3$, for calculating LP smoothing parameter $\theta = 30$ and time window parameter $\tau = 5$ is used. Furthermore, to determine the splitting dimension and value, 10 different split locations were used when computing the error rates of hypothetical regions. Results are compared with V-IAC by using the same parameters.

5.4. Results

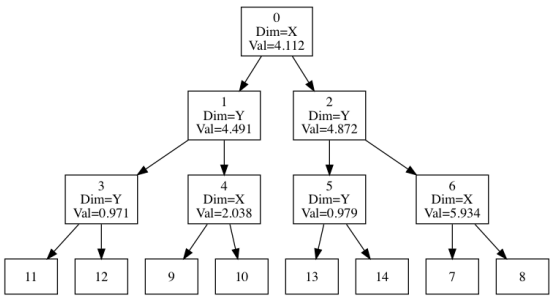
The proposed method is evaluated by analyzing the following: how well the autonomously generated sensorimotor regions reflect the underlying experimental setup (Subsection 5.4.1); whether our system allowed efficient and effective learning by analyzing the change in LP (Subsection 5.4.2), and the decrease in total error rate (Subsection 5.4.3). The proposed method is compared with V-IAC in our analysis.



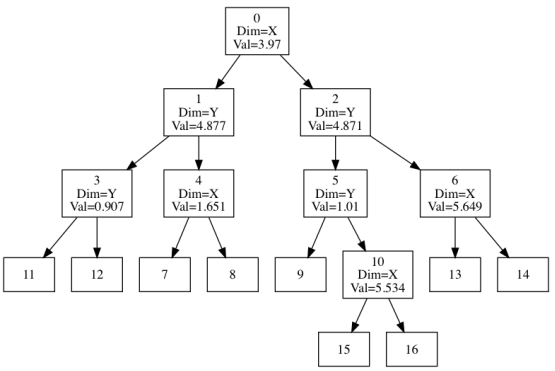
(a) $\mathbb{SM}$ tree formed by PE-IAC



(b) $\mathbb{SM}$ tree formed by PE-IAC



(c) $\mathbb{SM}$ tree formed by V-IAC

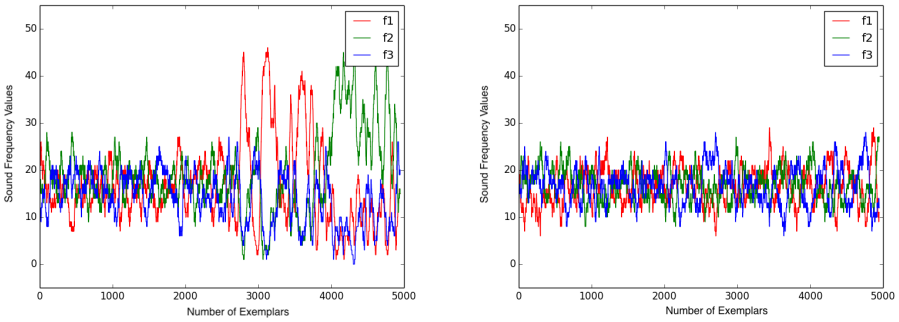


(d) $\mathbb{SM}$ tree formed by V-IAC

Figure 5.2. $\mathbb{SM}$ region tree formed by PE-IAC and V-IAC. Here first, second and third line corresponds to the ID, cutting dimension and cutting value of the region respectively. Arrows show the parent-child relationship between the regions.

5.4.1. Generated Sensorimotor Regions

Our aim in this method is splitting $\mathbb{SM}$ from the points which split the space in a semantically sensible and useful manner to improve the learning rate of the system. In order to evaluate this, both PE-IAC and V-IAC were trained with a 5000 iteration set. Due to the randomness, after a large number of runs, various distinctively structured trees were formed. Figure 5.2(a) and Figure 5.2(b) shows two representative trees

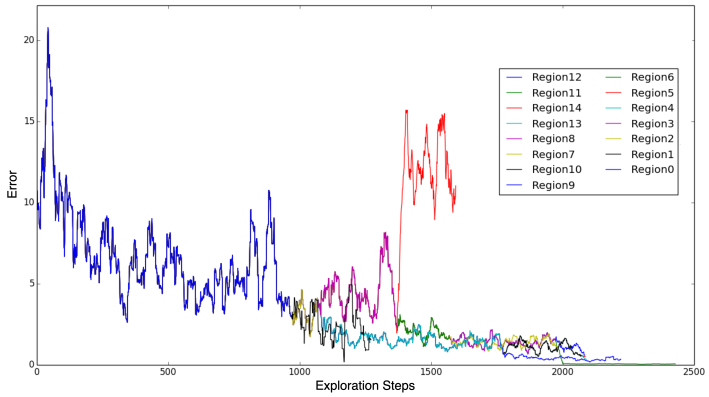


(a) PE-IAC (b) V-IAC

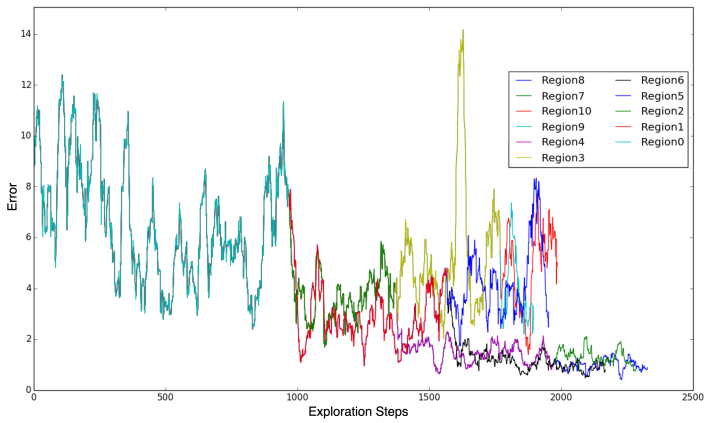
Figure 5.3. Sound frequency preferences in one run. The x-axis represents the number of exemplars collected by the agent and y-axis represents the sound frequency values obtained by considering and smoothing all the frequencies inside a time window.

formed during our experiments. In Figure 5.2, different trees produced by PE-IAC and V-IAC with the same parameters are shown. In Figure 5.2(a), the calculated cutting dimension of the 4^{th} region is compatible with the environment dynamics. The 6^{th} region covers A1 entirely, and the following splits are based on the f parameter, i.e., the system represents the effects of the f action parameter qualitatively. Another tree formed by the PE-IAC method is shown in Figure 5.2(b). In this tree, the first split occurred at $x = 0.805$. In this case, the system prefers exploring the region with x parameter larger than the cutting value 0.08, which is a value close to the boundary (1.0) that separates A1 and A2. Next, the 3^{rd} region is split by $y = 1.147$, which defines the other border between A1 and A3; then, there was no observation of further exploration in region 5. This is because the region is a non-learnable region since it involved randomness. As can be seen in both two children of the 2^{nd} region, the system was successful in terms of splitting the region by f parameter from the points which make a difference. As shown in Figure 5.2(c) and Figure 5.2(d), V-IAC could not split $\mathbb{SM}$ successfully taken into account the environment dynamics i.e., there were no trees formed by this method using f parameter as a cutting dimension.

We further analyzed how often which sound frequencies were explored by the robot for training different experts. When we compare plots in Figure 5.3, we observe



(a) PE-IAC



(b) V-IAC

Figure 5.4. Smoothed derivative curves of errors of PE-IAC and V-IAC. For visualization, smoothed error curves are represented according to robot's encountering time of the experiences.

that our method prefers using specific sound frequencies as time passes. However, such distinction in V-IAC is not discovered within 5000 iterations. Furthermore, it is observed that after the 10000 iteration phase, this parameter has still not been discovered in V-IAC. Thus, V-IAC could not explicitly identify and represent the effect of sound frequencies on the interaction of the robot with the environment.

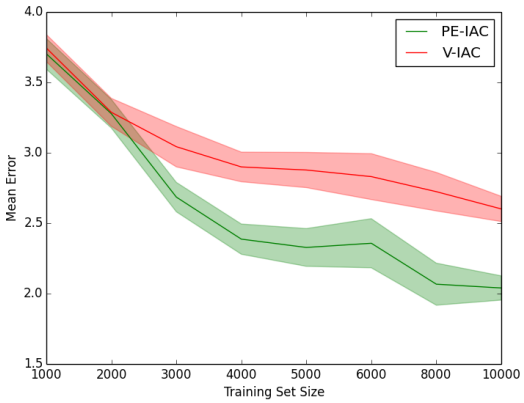


Figure 5.5. Comparison of the mean error of PE-IAC and V-IAC methods. The lines correspond to the mean error values of the test sets and shaded areas represent the variance.

5.4.2. Comparison of Learning Progresses of Regions

Here, we analyze the LP in each region and compare the LPs observed in PE-IAC and V-IAC. In Figure 5.4(a), error curve of each child region is preceded by that region's parent's error curve; thus, the decrease in the error after the split can be observed from the plot. First of all, as $\mathbb{SM}$ is split, the coherency of predictions is improved in general. However, the error rate of region 5 remains the same (and high) as it corresponds to A3 that is completely random and, therefore, it is not learnable. The smoothed derivative error curves generated by V-IAC are shown in Figure 5.4(b). In this method, for each child region, errors are completely inherited from the parent region, and error change can be observed by considering this case. When the two methods are compared, it can be observed that with our method error drops quickly in all regions except the unlearnable region and, with V-IAC, while error drops in some regions, it remains the same in other regions since those regions include parts from unlearnable A3.

5.4.3. Comparison of Decrease in Total Error Rate

For both methods, error rates of the systems that involve experiments with different training sizes are shown in Figure 5.5. For this plot, we have trained both methods with the number of exemplars stated in the x-axis. Next, both methods were tested

with 10 different test sets, each of them including 2000 exemplars. Initially, there is no difference in performance between the two systems in the training set with 1000 training exemplars, because the first split had not yet occurred, and the system has not made action selection by using its own IM. After training the system with 2000 exemplars, the gap between the performances of the methods becomes more apparent.

5.5. Conclusion

In this study, we have proposed a novel method for splitting $\mathbb{SM}$ to improve the learning performance and compared it with an existing approach. Our contributions in this study are (1) we have brought in a new splitting mechanism which considers the successes a broad range of potential learning regions and (2) while evaluating the LPs that is used as the intrinsic reward of the agent, we consider only the errors made by that learning region. As a result, the IM mechanism provided in our work splits $\mathbb{SM}$ more accurately and considers the future aspects of the splitting decisions. We have evaluated our method in a simple experimental setting; we have observed that the proposed method performs better than the existing method. Also, our method is more memory efficient since the child regions do not inherit error list of its parent. However, considering a broad range of candidate regions and re-evaluating their performances for each split decision, brings high computational overhead. Further, in this work, we set the total exemplar number that the agent will encounter during its lifetime. However, this mechanism should be autonomous, as well. After some point, the splitting mechanism should be stopped when the agent acquires a sufficient level of competency.

In this study, we considered partitioning the sensorimotor space by the potential prediction performance of the candidate regions. If we consider the interaction between the agent and the environment, this approach has a limitation about not considering the outcome of the interaction. In our next study, we address this limitation by utilizing a latent space that combines the outcome, action, and object-information. Our next study is verified with a setup that involves 3D interactions with a simulated robot.

6. PROPOSED SYSTEM

In this chapter, the proposed system is introduced, and the building blocks of the system are explained. The organization of the chapter is as follows: first, the representations of the state, action, and effect are explained in Section 6.1; next, an overview of the system and general workflow are given in Section 6.2, finally the structures of the individual components of the proposed system are presented in Section 6.3 - 6.6.

6.1. Representations of State, Action and Effect

In the proposed system, in order to build the representation of the state (the yellow box in Figure 6.1), the robot first executes randomly sampled actions in randomly sampled states. Here, the state (S) corresponds to the encoded depth image (I_{enc}) of the object that the robot is interacting with. In order to obtain a lower-dimensional encoding of the depth image (I), the output of the encoder part of a convolutional autoencoder is used. The encoder part of the convolutional autoencoders takes 128×128 depth image as the input and outputs 8D encoding of the given image. Although we tried lower-dimensional encoding sizes, using 8D as the image encoding provided better reconstruction performance.

The action (A) parameters comprise of the trajectory length (r_{path}), approach direction (ϕ_{path}), and the type of the action primitive. The end-effector of the robot follows a semi-circular trajectory, and r_{path} is the radius of this semi-circle, ϕ_{path} is the direction that the robot approaches the object. The type of end-effector aperture, namely *open*, *half-open*, and *closed*, determines the configuration of the end-effector. Therefore the action is a 5D vector. Note that, the end-effector of the robot always follows a semi-circular path independent from the configuration. The robot does not know about the semantic meaning of these end-effector aperture types; it only changes the pose of its fingers depending on the configuration input. These configurations are hard-coded; the robot does not make any alignment.

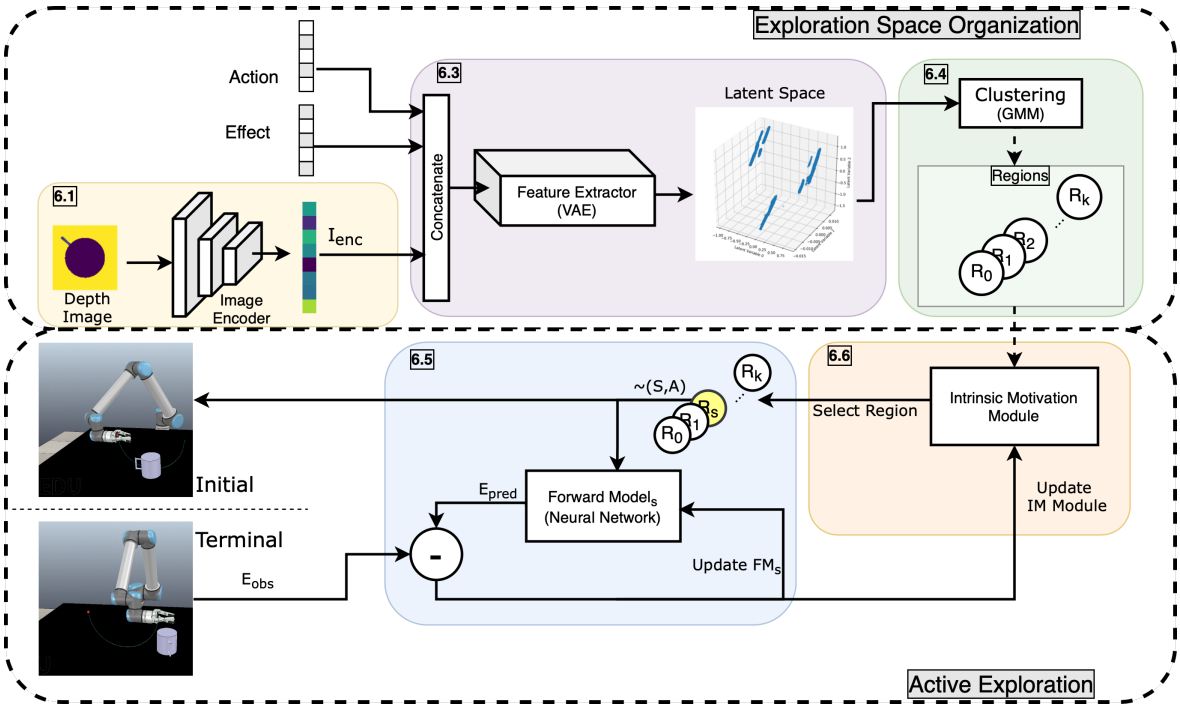


Figure 6.1. Structure of the proposed system.

Given the state and the action, the FM predicts the effect of this interaction on the object. The object to interact is always on the center of the table, i.e., its initial position does not change. The effect (E) is observed by executing the given action at the given state and comprises of the change of the x, y, z position, and the orientation around the z axis. To ensure the continuity of the orientation, sine, and cosine values are used. Thus the effect is a 5D vector. Further details about the state, action, and effect are given in Chapter 7.

6.2. Overview and General Flow

To make an agent adapt to different kinds of environments and dynamics and achieve tasks without being programmed to, it needs to be equipped with the ability to learn autonomously. For example, the environment may require interactions with different objects, different actions in various environmental conditions. We hypothesize that if the agent had a mechanism to identify these different characteristics, it would simplify the problem of learning the environment dynamics. To learn these different characteristics of the environment in a modular fashion, we consider identifying these

regions in a space that blends state, action, and effect information. Ideally, this space should not be dominated with the information that comes from one of the modalities (i.e., object, action, or outcome); it should highlight the most relevant links between the causes and the consequences. Therefore, we propose an approach with two stages: (1) organization of the exploration space, (2) active exploration. Note that it is assumed that the exploration space does not change during the active exploration stage.

In order to start organizing the exploration space, the initial interaction data of the robot is collected. From the interaction data, state, action, and effect vectors are concatenated and then used as the input of a feature extractor. The output of the feature extractor is considered as the latent space (Figure 6.1, the purple box). Next, the latent space is clustered, and the exploration regions are formed (Figure 6.1, the green box). After this stage, the formed regions are frozen, i.e., learning the characteristics of a region does not further alter its representation and the structure of the regions.

After forming the regions, the active exploration stage (Figure 6.1, bottom) begins. The agent learns the characteristics of the environment by exploring the regions. Each region has an exclusive forward model (FM) that is in charge of learning the characteristics of its region (Figure 6.1, the blue box). When a region is selected for exploration (R_s), a state and action pair (S, A) is sampled from the region, and the FM of the selected region predicts the effect (E_{pred}). Remark that we are using the latent space for only creating distinctive exploration regions. Sampling inside the region is made from the original data points, neither from the encoding nor from the reconstruction. The exploration strategy of the agent is regulated by the intrinsic motivation (IM) module (Figure 6.1, the orange box). During the exploration phase, the IM module selects the region (R_s) with the maximum learning progress signal. Ideally, the dynamics of the selected region is neither too easy nor too complicated for the agent's current skill set.

6.3. Formation of the Latent Space

To obtain a representation that includes the necessary and sufficient information about the link between the changes and its causes, we consider learning a latent space that blends the object, action, and outcome information. This information is extracted from the concatenation of S, A, E $(8D - 5D - 5D)$ (Figure 6.1, the purple box). From the computational perspective, it is desirable to have a compact representation that retains the information in the original dimensions efficiently and flexibly. To this end, a Variational Autoencoder (VAE) with Gaussian prior is used. Any feature extractor should work in practice; the choice of VAE is not central in our framework.

Following the input layer (18D), the encoder part of the VAE has an intermediate layer (9D) with ReLu non-linearity, followed by the hidden layer (3D) that is split into $\mu(z)$ and $\sigma(z)$ to allow reparameterization trick [35]. The decoder part has a structure that is symmetrical to the encoder part. It takes z that is sampled from the output of the encoder and has an output layer with sigmoid non-linearity. Binary cross-entropy is used as the reconstruction loss, and the VAE loss is calculated as in [35]. The VAE is trained with Adam [94] optimizer with a batch size of 100. The latent space is formed by using the $\mu(z)$ from the output of the encoder. In order to form the latent space, 10% of the total interaction data is used. The procedure that explains the division of the latent space into regions with different characteristics is explained in the next section.

6.4. Formation of the Exploration Regions

Before starting the active exploration, the separate characteristics of the environment need to be identified. Assuming that the latent space formed in the previous step has the necessary representation capacity for such an identification, the separation of the environment characteristics is accomplished by clustering the latent space. Each cluster is called a "region" of the environment. As the clustering algorithm the Gaussian Mixture Model (GMM) is employed. This selection is due to its flexibility to define the covariance shape and its probabilistic nature.

Assuming that the clusters have the same shape, tied covariance is used. For the experiments, per-sample average log-likelihood scores for [2, 20) of Gaussians are examined. By applying the *elbow method* to these scores, the number of Gaussians to be used is determined.

6.5. Prediction Models

To be competent in a given environment, an agent must have the ability to predict the consequences of its actions. Following [14, 15], instead of learning the overall dynamics with only one large and complex model, we consider dividing the effect prediction task into smaller subtasks that can be learned by simpler models. Each region has an associated FM that is responsible for predicting the effect given a state and an action. The forward models are implemented as feed-forward neural networks with input layer (13 units), one hidden layer (512 units) of Rectified Linear Unit (ReLu) non-linearity, followed by the output layer (5 units). Each FM is trained by backpropagation of the prediction error calculated as the mean square error (MSE), and trained by using Adam [94] optimizer. For each exploration step, an FM is trained with all the data which it countered so far. In order to prevent overfitting, the FM of the chosen region is trained for 5 epochs at each step.

6.6. Intrinsically Motivated Learning Module

To learn about the characteristics of the regions in the absence of an external task definition, the agent should have the ability to do autonomous exploration. In our study, autonomous exploration is driven by intrinsic motivation. In this section, the initialization of the intrinsic motivation module and its mechanism are explained.

In our study, we utilize "learning progress" as the intrinsic motivation signal. The learning progress measure gives the prediction performance change of a FM between consecutive time windows. At each step of the exploration phase, the learning progress measures of the regions are calculated to select the region to proceed. The learning progress signal is calculated by the approach suggested by [13].

In order to start active exploration, the system needs initial values for the learning progress values of the regions. The initial values for the learning progress are collected from a short bootstrapping phase. In this phase, each FM samples a predefined number (K) of state and action pairs. Then starts the training by taking smaller batches (κ) and records the error made on the sampled set. After collecting K/κ number of error records, the exploration phase begins. Note that the exploration phase begins after each FM finishes the bootstrapping phase.

Throughout the exploration, each region keeps track of its error list from the beginning. At time t, the error of n^{th} region ($e_n(t)$) is calculated by the MSE between the predicted effect E_{pred} and the observed effect E_{obs}. Learning progress of n^{th} region's forward model ($\mathrm{FM_n}$) at time t is calculated by taking mean error (γ_n) differences of most recent errors as follows:

$$LP_n(t+1) = \gamma_n(t+1-\theta) - \gamma_n(t+1) \tag{6.1}$$

Mean error of $\mathrm{FM_n}$ (γ_n) at time t is calculated as follows:

$$\gamma_n(t+1) = \frac{\sum_{i=0}^{\theta} e_n(t+1-i)}{\theta+1} \tag{6.2}$$

where the window parameter (θ) is empirically set to 16. The window parameter (θ) allows the system to capture the trend of the errors by averaging them within a time window and prevents the fluctuations from affecting the IM signal.

The intrinsic motivation module selects the region to explore by considering the learning progress of the regions. The region to be explored is selected using the ϵ-greedy [84] selection mechanism.

7. EXPERIMENTS

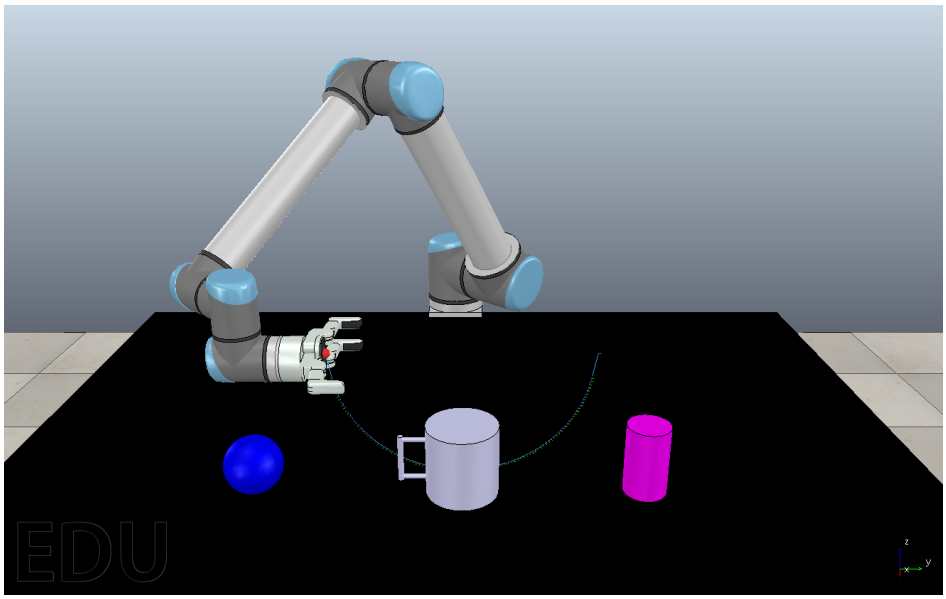


Figure 7.1. The experiment setup. A manipulator robot interacts with one of three types of object on a table-top environment. While the interaction, only one of these objects is present on the table.

The experiment setup is simulated in CoppeliaSim [39]. It consists of a six-degrees-of-freedom manipulator robot (UR10 [36]) that interacts with one of three different types of objects (cup, cylinder, and sphere) using its three-finger gripper as shown in Figure 7.1. The robot is capable of following a parameterized trajectory with a predetermined shape. While following the trajectory, it may use one of the three end-effector aperture width configurations that enable single-object interaction. For each interaction, the simulation scene is reset, and the parameters of the selected type of object and actions are sampled from a predefined range of values.

7.1. Parameterizations

7.1.1. Objects

The cup has a fixed height and radius of $(h, r) = (15, 7.5)$ cm and has a handle that is 12.5 cm apart from the center with a length of 10 cm. The orientation of the

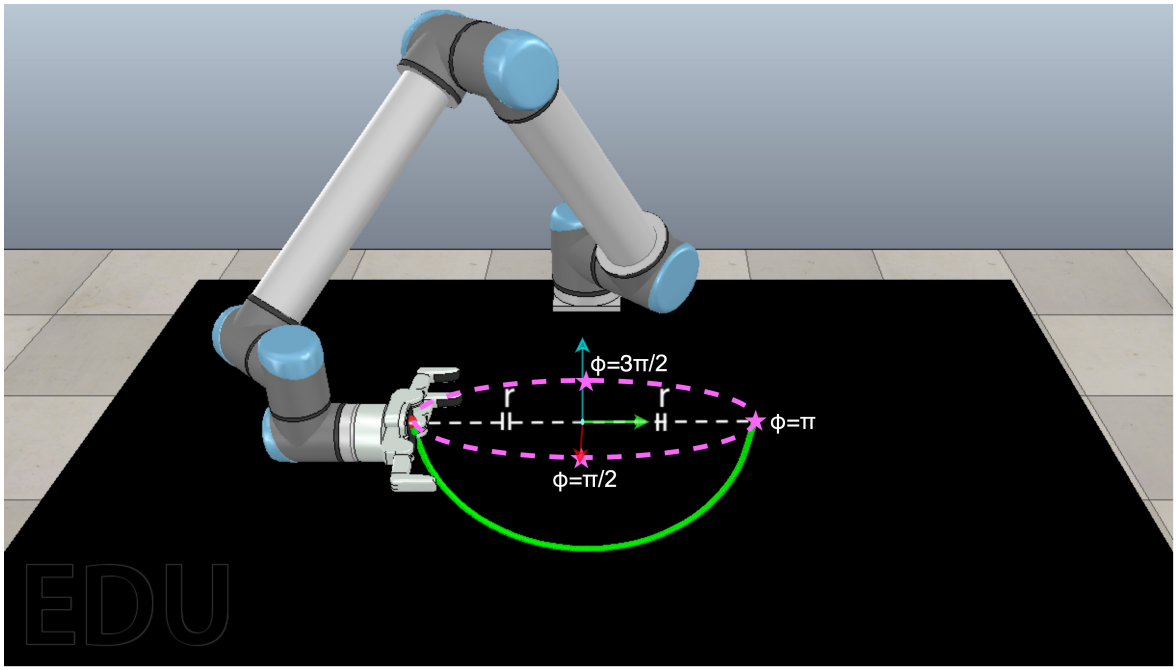


Figure 7.2. The end-effector follows a semi-circular path above the surface of the table. The action parameters controls the radius of the semi-circle (r_{path}) and the approach position of the end-effector to the object (ϕ_{path}).

cup around the normal vector from the surface of the table varies within $[0, 2\pi]$ among the experiments. Cylinders have a fixed height of $h = 15$ cm and radius within the range of $[1.5, 7.5]$ cm, and spheres have a radius within the range of $[3, 7.5]$ cm.

To encode the depth image of the objects into state, the encoder part of an autoencoder is used. The encoder consists of stacks of a convolutional layer followed by batch normalization and max-pooling operations, with channel numbers 512, 256, 128, 64, 32, 16 and 8. The decoder part uses up-sampling instead of max-pooling and has sigmoid non-linearity at the output layer. The rest of the decoder is symmetrical to the encoder part. In order to ensure that the lower dimensional state representation preserves the object-related information in the original dimensions, the quality of the reconstruction by the decoder part is considered. The autoencoder is trained by using binary cross-entropy as the reconstruction loss and Adadelta [95] optimizer.

7.1.2. Actions

The end-effector of the robot follows a semi-circular path that has start and end-points with the same elevation from the tabletop. The closest point of the path to the table is the halfway point, and it has a fixed offset from the surface of the table to avoid collision between the end-effector and the table. The semicircular path is defined with $r_{path} = [26, 31]$ cm with a z-orientation within $\phi_{path} = [0, 2\pi]$ radians. ϕ_{path} controls the approach direction of the end-effector to the object, i.e., controls the position of the end-effector on the magenta ellipse around the cyan arrow shown in Figure 7.2. The action parameters are $A = (r_{path}, \phi_{path}, closed, half - open, open)$ including the radius and z-orientation of the path and a one-hot encoded vector that encodes the type of the end-effector aperture respectively. The *open* end-effector configuration is implemented similar to grasp-reflex, i.e., when an object is inside of the end-effector, the robot closes the fingers. The object is picked up by *half-open* configuration if the object is between two parallel fingers. The *half-open* configuration is realized by adhering all three fingers in front of the palm.

7.1.3. Effects

The effect is defined as the change of the position and the z-orientation of the object $E = (\Delta x, \Delta y, \Delta z, \overrightarrow{\Delta\phi_z})$ after applying the action. The effect is calculated by taking the difference between the first and final pose of the object. We used sine and cosine values of $\Delta\phi_z$ to ensure continuous change. Note that, even if the robot executes the action with open and half-open end-effector aperture configurations to objects, it may not elevate the object. This is because of the object size, object pose, and end-effector pose misalignment and simulation noise. For example, if the robot approaches with an open end-effector to the cup from the side of the cup's handle, due to the contact of the handle with fingers, the object rotates and is pushed out of the end-effector; hence it can not be grasped and elevated.

7.2. Data Collection

At the beginning of each interaction, an object among one of the object types with its corresponding parameterization is created in the center of the table. A Kinect [37] camera is positioned on top of the table and records 128×128 depth image (I) of the table-top. Then the robot changes the pose of the gripper depending on which action primitive to use given within the action parameter vector, and the end-effector follows the semi-circular path that is tangential to the surface of the table center using the inverse kinematic module of CoppeliaSim. The effect is recorded when the end-effector reaches the end of the path, and simulation is reset. All the parameters are sampled uniformly from their corresponding intervals. Overall, the dataset consists of three different object types with three different action types that each consist of 5184 interactions, in total $3 \times 3 \times 5184 = 46656$ interactions.

8. RESULTS

In our study, we hypothesize that forming a latent space that blends the object, action, and outcome information allows identifying the diverse characteristics that reside in the environment. In the rest of this thesis, the method that forms the regions by clustering the latent space will be called as LatentIM. To validate our hypothesis, we consider three different strategies to identify different characteristics of the environment: The methods forming the regions by clustering effect, state, and action spaces will be referred as EffectIM, StateIM, and ActionIM respectively. All of these methods use the same clustering algorithm with the same parameters. However, the space that the clustering is applied differs. Finally, the baseline for the abovementioned strategies is the RandomIM, which assigns regions to the data points randomly. Note that, for all these strategies, the active exploration phase is identical, i.e., they only differ in the formation of the regions.

We conducted experiments to answer the following questions:

(i) How does the method of region formation affect the overall prediction performance? (Section 8.2)
(ii) Does LatentIM create an ordering between different skills in the latent space? (Section 8.3)
(iii) What is the effect of the different hyperparameters (number of clusters, ϵ, exploration strategy) on overall performance? (Section 8.4)

Note that unless stated otherwise, we used image encoding (I_{enc}) with a dimension of 8 as the state, the number of samples included to the training of each FM in during the bootstrapping phase is $K = 128$, for each exploration step the selected FM takes $\kappa = 16$ samples from its region, $\epsilon = 0.3$, $\theta = 16$, and the number of regions is 5. All the FMs are neural networks with one hidden layer of 512 units with ReLu activation function.

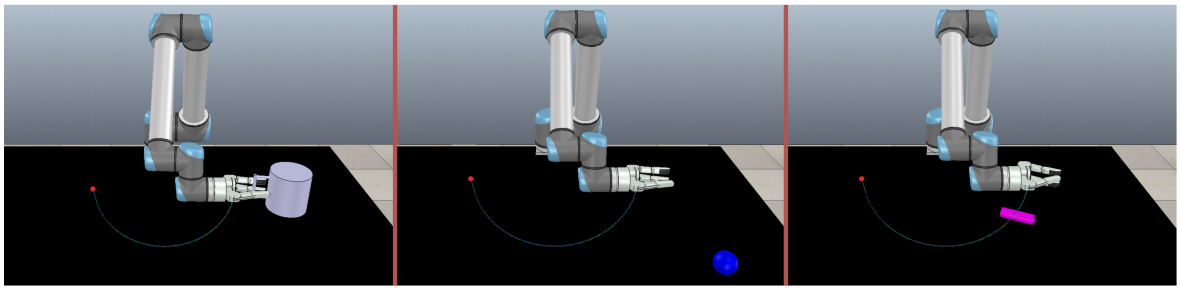


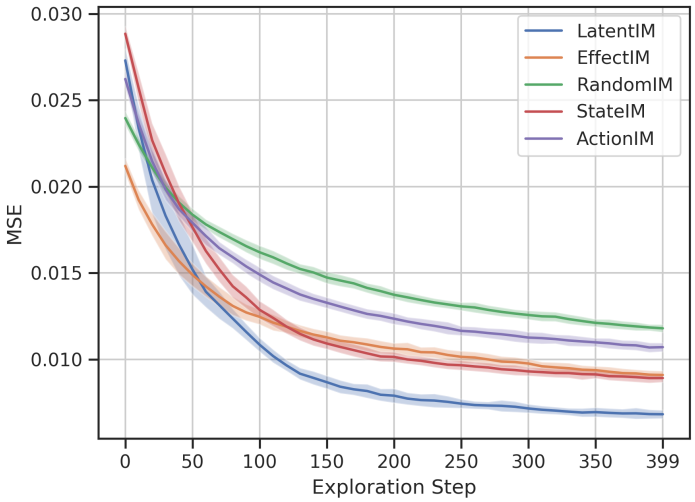
Figure 8.1. Final states of several executions. The left, middle and right columns show lifting the cup, pushing the sphere, and tumbling the cylinder, respectively.

8.1. Skills Developed by LatentIM

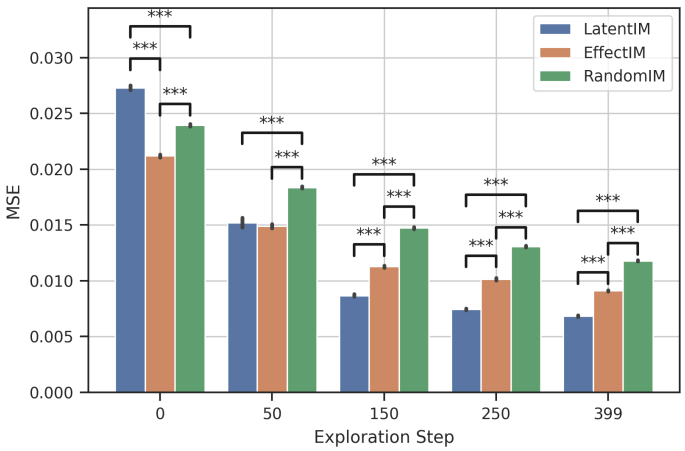
By checking the clusters that are created by clustering the latent space, we observed the following groupings: (1) the one that uses *half-open* end-effector configuration and has a uniform distribution in all features but $\Delta z = 0$; (2) the one that uses *open* end-effector configuration and has a uniform distribution in all features but $\Delta z = 0$ of the object pose; (3) the one that uses *half-open* end-effector configuration and has a uniform distribution in all features of the object pose; (4) the one that uses *open* end-effector configuration and has a uniform distribution in all features of the object pose; (5) the one that uses *closed* end-effector configuration and has a uniform distribution in all features. Considering the characteristics, in the rest of this thesis, we will call (1) as non-lifting pinch-grasp (*n_pinch*), (2) as non-lifting power-grasp (*n_power*), (3) as lifting pinch-grasp (*l_pinch*), (4) as lifting power-grasp (*l_power*) and (5) as non-lifting (*n_push*). Note that these labels are given in order to help the reader; we did not use them in our experiments. The developmental order of these skills are presented in Section 8.3.

8.2. Comparison of Overall Performances

In this section, we investigate the effect of the region formation on the overall prediction performance of the system. For the comparison, we consider five different mechanisms for the region formation, i.e., LatentIM, EffectIM, RandomIM, StateIM, and ActionIM.



(a) The change of weighted average MSE through exploration in $N = 40$ experiments. The shaded areas show the standard deviation.



(b) Statistical significance analysis for LatentIM, EffectIM, and RandomIM.

Figure 8.2. Comparison of the prediction performances of LatentIM, EffectIM, StateIM, ActionIM and RandomIM throughout the exploration.

Figure 8.2(a) shows the average weighted MSE of the $N = 40$ experiments. We probed the performances of methods on all data at every 10 exploration steps. Since the number of samples may differ between the regions, the mean MSE of one experiment is calculated by weighting the MSE of a region by its number of samples. Next, the mean and the standard deviation of the weighted MSEs are calculated for all experiments

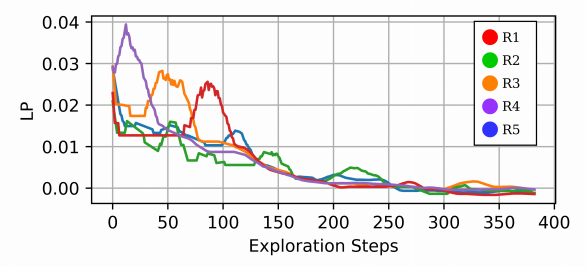
among all the probing time-steps. Note that the bootstrapping phase is not included in the plot in Figure 8.2(a). As presented in figure LatentIM gives the lowest error among all five cases. Following LatentIM, the EffectIM and StateIM perform similarly. The only difference we observed for those two is that the EffectIM is better at the beginning, but StateIM shows a more rapid decrease in the MSE. Similar states can result in significantly different effects depending on the actions applied, and similar effects may be observed with different states and actions. Following the EffectIM and StateIM, ActionIM gives a lower performance, since it does not include the state information to form the regions. Overall, the RandomIM performs the worst, and here we present it as a baseline.

In Figure 8.2(b), we present the statistical analysis of the differences between LatentIM, EffectIM, and RandomIM taken from the different exploration steps. We ran analysis of variance (ANOVA) to check whether the MSE distributions of these three approaches are different, then carried post-hoc ANOVA tests, i.e., Tukey's HSD test and Games-Howell test, depending on the equal and non-equal variance cases, respectively. We found that after $t = 50$, the performance of LatentIM and EffectIM differs significantly $p < 0.001$, LatentIM giving more accurate predictions.

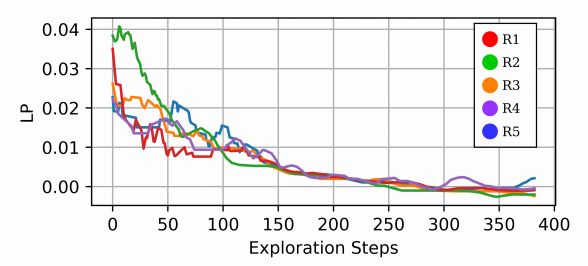
8.3. Developmental Order of Skills

In this section, we analyze the developmental order of the skills that is regulated by the intrinsically motivated exploration. The analysis of the developmental order observed in the single runs of LatentIM, EffectIM and RandomIM are presented in Figure 8.3, and average of 40 runs of LatentIM is presented in Figure 8.4.

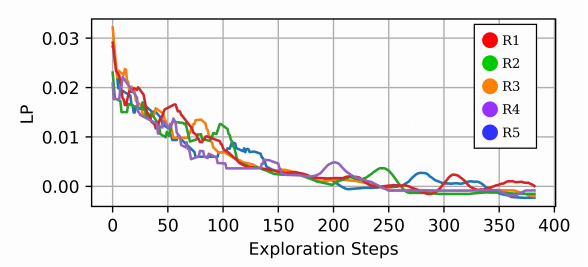
Figure 8.3(a) shows the change of the learning progress of the regions of LatentIM in a single run. The regions of LatentIM show a clear ordering between the skills that we have discussed in Section 8.1. It first explores the lifting grasps i.e., *l_power* (R4) and *l_pinch* (R3) then shifts its attention to *n_pinch* (R1), *n_push* (R5) and *n_power* (R2), respectively. Note that the order of skills may change across different runs, due to the randomness involved in ϵ-greedy region selection and sampling inside the regions.



(a) Change of LP in LatentIM



(b) Change of LP in EffectIM



(c) Change of LP in RandomIM

Figure 8.3. Changes of learning progresses of 5 regions during the exploration phase from a single run. The learning progress signal is smoothed by ($\alpha = 16$ window) to make it easier to perceive.

Detailed investigation and statistical analysis of the developmental order formed by LatentIM will be discussed later in this section.

In Figure 8.3(b), EffectIM shows a preference over $R2$, which corresponds to the data points that produce a change in x, y position and the orientation of the cup object, at the beginning but, following that, no clear ordering over the regions is observed. Likewise, $R4$ contains the same kind of object, action, and effect types. The only difference among those two is the change in the z-orientation of the object, i.e., $\sin(\Delta\phi_z), \cos(\Delta\phi_z)$. Similarly, in 8.3(c) there is no ordering between the regions for

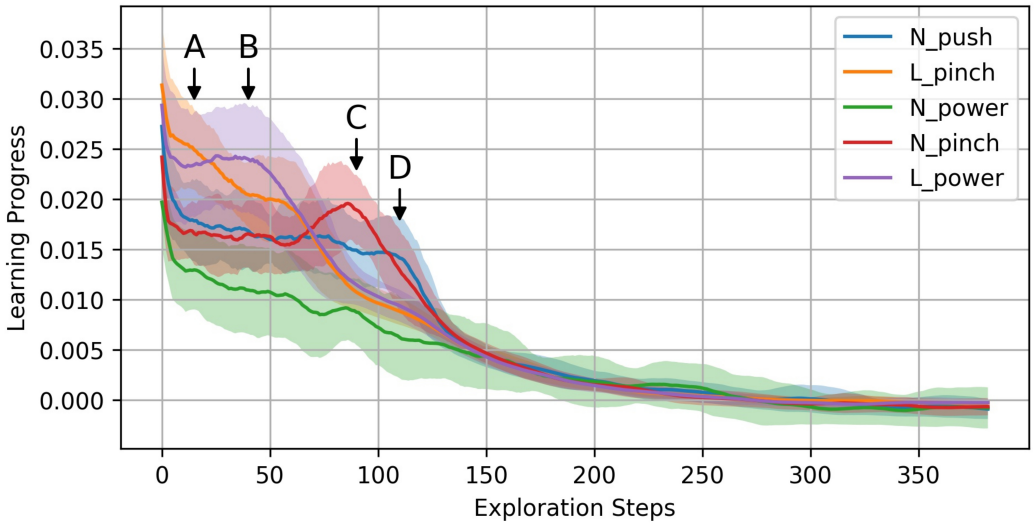


Figure 8.4. Change of mean learning progress of 5 LatentIM regions during the exploration phase. The plot is the smoothed by a time window ($\alpha = 16$). The lines show mean LP values and the shaded areas show the standard deviation.

RandomIM. Notice that through the end of the exploration phase, for all the strategies, the LP values reach zero, even slightly go to negative values. By considering the capacity of the predictor, this situation means that there is no advantage of further learning. It has already learned what can be learned from this region. Negative LP values are due to the fluctuations caused by the slight changes in the prediction errors.

Figure 8.4 shows the average ($N = 40$) change of the learning progress of the regions of LatentIM. The plot is created by calculating the mean and the variances of the regions, then smoothing by $\alpha = 16$. Since we are using neural networks as forward models, the training examples encountered by the FMs alter the learning progress values and their changes during the exploration. However, by looking at the averages, as shown in Figure 8.4, we observe a consistent ordering as *l_pinch*, *l_push*, *n_pinch*, *n_push* and *n_power*. This ordering is sensible because, when grasped, the z-orientation change is approximately zero. However, when pushed or not-lifted with pinch/power grasp, the object may turn, tumble and stray away from the end-effector's trajectory. Thus, when the robot lifts the object, the effect is more predictable than the rest and hence easier to learn.

To evaluate the significance of the ordering presented in Figure 8.4, A Kruskal-Wallis test was performed on the learning progresses of the five different regions. The differences between the learning progress distributions of the regions taken from the interval $t = [0, 139]$ were significant with $H(4)$, $p < 0.01$. Following that, we also performed the Mann-Whitney U test to determine the significance of the learning progress values for the pairs of regions.

In Figure 8.4, first *l_pinch* has the maximum learning progress value. Taken from that interval at *Point A* the LP of *l_pinch* was significantly greater than of *l_power*, $p < 0.05$ and the LP of *l_power* was significantly greater than the rest with $p < 0.001$. At the same time-step, *n_pinch* was not significantly different than *n_push*, while the LP of *n_push* being significantly greater than of *n_power*, $p < 0.01$. Following the plot, we see the dominance of *l_power* over *l_pinch*. Taken from that time window, at *Point B*, LP of *l_power* was significantly greater than of *l_pinch*, $p < 0.05$. While the LP of *l_pinch* is significantly greater than of *n_push* and *n_pinch* with $p < 0.001$, the difference between *n_push* and *n_pinch* was not significant, both being greater than *n_power* with $p < 0.01$. After the decrease of *l_pinch* and *l_power*, a significant increase in *n_pinch* is visible at the Figure 8.4. Being within that time window, at *Point C*, the LP of *n_pinch* was significantly greater than of *n_push*, $p < 0.05$. While LP of *l_pinch* is significantly less than of *n_push* with $p < 0.05$ it was significantly greater than of *n_power*, $p < 0.001$. And finally, there is a short primacy of *n_push* over the rest, *Point D*, the LP of *n_push* was significantly greater than of *n_pinch*, $p < 0.05$. In summary, our method follows this order among the regions: *l_pinch*, *l_power*, *n_pinch*, *n_push* and *n_power*.

8.4. Effects of Hyperparameters

In this section, we provide additional experiment results to examine the effect of the hyperparameters, i.e., ϵ, the number of clusters, and the exploration strategy.

Table 8.1. Mean MSE values for LatentIM, EffectIM and RandomIM with different ϵ values. Results reported here show the weighted MSE of the last exploration step for each condition.

	F-Score	μ(LatentIM)	μ(EffectIM)	μ(RandomIM)
$\epsilon = 0.0$	1680.05	0.007121	0.009238	0.011895
$\epsilon = 0.5$	3635.97	0.006753	0.009065	0.011748
$\epsilon = 0.7$	3873.38	0.006724	0.009093	0.011675
$\epsilon = 1.0$	4291.49	0.006819	0.008934	0.011766

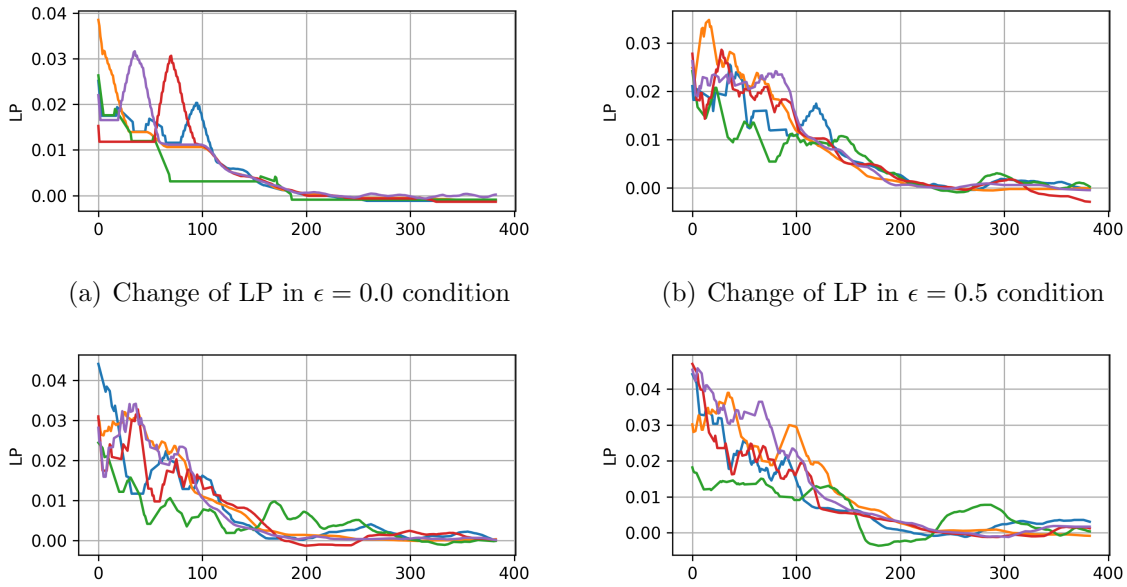
8.4.1. Effect of ϵ Parameter

In this subsection, the effect of the ϵ parameter on the overall prediction performance and the developmental order in LatentIM is analyzed.

In Table 8.1, we present the performances of LatentIM, EffectIM and RandomIM by using different values of the ϵ parameter. We conduct One-Way ANOVA $F(2, 87)$, and post-hoc Tukey's HSD. Pairs (LatentIM vs. EffectIM, LatentIM vs. RandomIM, and EffectIM vs. RandomIM) for each ϵ value. The test yield that each pair is different with $p < 0.001$.

As it is explained in Section 6.6, the ϵ parameter controls the ratio of exploration steps with random exploration to all exploration. For all the ϵ values, we observe that the LatentIM gives the lowest error among the other two. We conduct $N = 30$ experiments for each ϵ value and verify our results with One-Way ANOVA tests followed by Tukey's HSD post-hoc tests. We observe that except for $\epsilon = 0$, the performance of LatentIM does not change significantly.

In Figure 8.5 we present the single run learning progress evolutions of LatentIM with different ϵ conditions. Increasing values of ϵ prevents seeing an ordering between different skills and does not provide a further benefit for the predictor performance.



(a) Change of LP in $\epsilon = 0.0$ condition

(b) Change of LP in $\epsilon = 0.5$ condition

(c) Change of LP in $\epsilon = 0.7$ condition

(d) Change of LP in $\epsilon = 1.0$ condition

Figure 8.5. Learning progresses of the regions of LatentIM with different ϵ parameters. The plots are smoothed with $\alpha = 16$ time windows.

8.4.2. Effect of Number of Clusters

In this subsection, the effect of the number of clusters on the prediction performance is analyzed. Prior to the experiments, the per-sample average log-likelihood scores for $[2, 20)$ of Gaussians are examined. The performance of different numbers of clusters by applying the elbow method to these scores is presented in this subsection.

In Table 8.2, we present the performance comparisons of LatentIM, EffectIM and RandomIM with different number of clusters. For this experiment, we use the given hyper-parameters except for the number of clusters, which is the independent variable. The results show the mean and standard deviations of the methods' MSE calculated by conducting 30 independent experiments each. Each experiment take 400 exploration steps. To check the statistical significance of our findings, we use the One-Way ANOVA test followed by Tukey's HSD post-hoc analysis. We observe with $5, 6$ and 7 clusters, LatentIM gives a lower prediction error that is statistically significant with $p < 0.001$. In the Table 8.2, italic font is used to indicate the lowest mean MSE that is statistically significant with indicated α.

Table 8.2. Prediction performance of LatentIM, EffectIM and RandomIM with different number of clusters. The table presents mean (μ) and standard deviation (σ) of the errors in the last exploration step collected from 30 experiments.

# of Clusters	LatentIM		EffectIM		RandomIM		$\mu(L) - \mu(E)$	$\mu(L) - \mu(R)$	$\mu(E) - \mu(R)$
	μ	σ	μ	σ	μ	σ			
2	0.009876	0.000293	*0.009090*	0.000232	0.010308	0.000302	0.00079^{***}	-0.00043^{***}	-0.00122^{***}
3	0.009714	0.000354	*0.008502*	0.000231	0.011000	0.000210	0.00121^{***}	-0.00129^{***}	-0.00250^{***}
4	0.008451	0.000352	0.008391	0.000203	0.011424	0.000258	0.00006^{ns}	-0.00297^{***}	-0.00303^{***}
5	*0.006836*	0.000194	0.009195	0.000255	0.011787	0.000199	-0.00236^{***}	-0.00495^{***}	-0.00259^{***}
6	*0.008892*	0.000260	0.010928	0.000264	0.012116	0.000186	-0.00204^{***}	-0.00322^{***}	-0.00119^{***}
7	*0.007666*	0.000291	0.010956	0.000231	0.012588	0.000186	-0.00329^{***}	-0.00492^{***}	-0.00163^{***}

8.4.3. Effect of Different Exploration Strategies in Latent Space

In this subsection, we analyze the effect of the exploration strategy on the overall prediction performance. For this experiment, we consider two different strategies, along with our LP-based intrinsically motivated exploration strategy, which will be referred as IME in this subsection. Following the strategies presented in [21], we implemented Fixed Curriculum (FC) and Single Goal Space (SGS). While the former selects regions in turn according to $exploration\_step \ \% \ \#of\_regions$, the latter always selects only one region throughout all the experiments.

Since the regions are formed by clustering the latent space, in the first step of the exploration performances are not significantly different. This is an expected result since, during the bootstrapping phase, each of them undergoes from the same procedure, i.e., all the FMs for each strategy samples an equal number of data-points from their corresponding region and trains the forward model before moving on the exploration phase. Throughout the exploration, we see that the SGS stops improving itself; it is due to the convergence of the selected region's forward model.

For these strategies, we analyze the prediction performance on different output channels at the end of the exploration phase by conducting Mann-Whitney U tests. A Mann-Whitney test indicated that the prediction error of Δy is less than for IME

($\mu = 4.281\,cm$) than for FC ($\mu = 4.437\,cm$) cm, $p < 0.01$, both being less than SGS ($\mu = 17.986\,cm$), $p < 0.001$. Later, we calculated the $\Delta\phi_z$ from the sines and cosines predictions and found that, IME ($\mu = 8.0567°$) is significantly less than FC ($\mu = 8.3312°$), $p < 0.001$ both being significantly less than SGS ($\mu = 8.7922°$), $p < 0.001$.

9. DISCUSSIONS

9.1. The Developmental Order Created by LatentIM

In Figure 8.4 we observed that the *l_pinch* is explored before *l_power* (see Section 8.1). However, pinch grasp requires more precise control and exact movements and develops after the power grasp in infants [1]. Considering the limited capabilities of the simulators, simulating these precise movements and the inherent noise of the world is intricate. Therefore, we consider this is a natural consequence of using a simulator.

As discussed by Oudeyer *et al.* [13], the complexity of tasks is observer-dependent, and comparing the developmental order with infants is introducing an anthropomorphic bias. Since it is almost impossible to mimic biological, physiological, and psychological aspects of the development of infants in robots, such an ordering can be explained better with the distribution of data. We have discussed the data distributions of LatentIM in 8.1. We did not observe a similar ordering in comparison methods because they did not create regions with such data distributions. Still, the clustering algorithm and the parameters had to be the same for all these methods to provide a basis for comparison. We investigated the effect of the number of clusters in Section 8.4.

Redesigning the experiment setup in order to support tool use would be an interesting direction for our research. It is known that the tool use abilities of infants develop after becoming competent in basic manipulation skills [96]. Such an experimental design for intrinsically motivated exploration can be seen in the study presented by Forestier *et al.* [21].

9.2. Change in LP vs. Developmental Order

In vertebrates, the basal ganglia manage the selection between the competing motor programs and cognitive resources [97]. When an organism needs to attend more than one system, it needs to select one of them. Indeed, this is a resource

allocation problem. In computer science, a vast number of heuristics and architectures are developed for selecting and scheduling between competing resources. Redgrave *et al.* [97] state that the selection mechanism should consider the influences of individual competitors, should provide an order of priority among them, and should be adaptable. Thus, considering the intrinsic motivation signal, which is the learning progress in this study, provides the robot a heuristic to select between the competing resources (i.e., exploration of regions). Considering the "winner-take-all" strategy, as the learning progress of the selected region falls below the second candidate, switching occurs, and it continues like this [97]. Readers referred to the paper of Mirolli *et al.* [98] for a review about the relationship between dopamine and prediction error signals.

9.3. Fixed Representation Assumption

A potential limitation of our work is that utilizing a fixed representation throughout the exploration phase. Piaget states that the cognitive development of children is a continuous improvement of *schemas* [99]. He argues that development stems from biological maturation and interaction with the environment [100]. As children explore, through *assimilation* and *accommodation*, they update the existing mental representation of the world [101]. However, our fixed representation assumption is only for implementation purposes. We do not have any claim for the developmental progression of the mental representations.

9.4. Object-based Region Emergence

Considering the formation mechanism of the exploration regions, we also expected to see an object-based region emergence. Failure to observe such a situation can be explained by several reasons such as (1) selection of input and output representations, (2) experiment setup, (3) mechanisms that form the latent space and (4) image encoding procedure. Regarding (1), including distinctive features to the effect may result in an object-based emergence. For example, the inclusion of x and y-orientations ($\overrightarrow{\Delta\phi_x}, \overrightarrow{\Delta\phi_y}$) to the effect features would create distinct effects with cylinders and cups than the spheres. Since a sphere can not be tumbled, we could observe a separate

region. Concerning (2), using different objects and action parameterizations that can create more diverse representations could support such an emergence. Even if the depth image is used to obtain state representation, three types of objects look similar, considering the position of the vision sensor. For example, using a box would result in more distinct encodings. On (3), fine-tuning of the VAE parameters and using more complex priors might cause observing distinctive object-based regions. About (4), although using 8D image encoding (I_{enc}) provides a good reconstruction performance, since it is not jointly learned with the latent space, it may not be able to find such a distinction. We had to omit such a design due to the limited computational resources.

9.5. Utilization of Latent Space

In our study, we used VAE to form the latent space to separate the regions with different characteristics. However, we sampled corresponding state and action pairs from the original versions of these data points, i.e., we did not use the reconstruction or encoding created by VAE. In the literature, there are interesting applications with VAE, such as producing fake human faces. Considering the generative nature of the VAEs, they could be utilized to create various learning examples. Such a property can lead to dataset augmentation similar to [88], and real robot experiments could benefit from it.

9.6. Observed Error and Sensory Noise

In our study, we have used the change of the position and orientation of the object obtained from the simulator as the effect. However, in the real world, the effect of such interactions is obtained from visual and tactile sensory information. In the grasp case, the effect is obtained by fusing the information coming from tactile and visual sensory channels. In contrast, in the push case, the effect can only be obtained from the visual sensory data. If we had used tactile and visual information instead of using the data provided by the simulator and introduced different sensory noise levels, it would allow us to see whether the push case is more difficult to predict than the grasp case in real-world scenarios. Since the effect observed by using only visual channels would be

noisier than the combination of the visual and tactile channels, the usage of sensory data for determining the effect might change the performance and the order exhibited by the system depending on the level of the sensory noise. Therefore, it would be an interesting direction for our research to use sensory data for making effect prediction.

10. CONCLUSION

In this thesis, we studied intrinsically motivated exploration strategies that enable artificial agents to discover skills and acquire knowledge about the environment autonomously. Inspired by the intrinsic motivation concept observed in humans and animals, many developmental robotics and machine learning studies considered utilizing computational models of intrinsic motivation to provide (1) an intelligent exploration scheme for efficient sampling of the learning examples, (2) a way to deal with sparse reward settings in RL scenarios, (3) a detection mechanism for salient events in the environment. In this study, we employed intrinsically motivated exploration to enable efficient and effective learning of environment dynamics to discover a set of skills.

In our first study, we utilized an existing intrinsic motivation framework [13] and proposed a mechanism to partition the sensorimotor space. In that work, during the exploration, the sensorimotor space is continuously partitioned into regions by considering the potential prediction performances of the candidate regions. By evaluating the method in a simple 2D experiment setup, we found that this partitioning scheme enabled the discovery of semantically meaningful regions in terms of environment dynamics and presented a better prediction performance.

In our subsequent work, we propose utilizing a latent space that blends outcome, action, and object-related information to discover distinctive interaction schemes between the robot and the environment. These schemes are learned by separate forward models that predict the outcome given the state and the action. By determining the exploration regions, we have shown that these regions correspond to semantically meaningful regions that combine action and outcome information. In our experiments, we compare our proposed method with the methods that consider different modalities to form exploration regions and show that the proposed method exhibited a better prediction performance.

REFERENCES

1. Cangelosi, A. and M. Schlesinger, *Developmental robotics: From babies to robots*, MIT press, 2015.

2. Oudeyer, P.-Y. and F. Kaplan, "What is intrinsic motivation? A typology of computational approaches", *Frontiers in neurorobotics*, Vol. 1, p. 6, 2009.

3. White, R. W., "Motivation reconsidered: The concept of competence.", *Psychological review*, Vol. 66, No. 5, p. 297, 1959.

4. Berlyne, D. E., "Curiosity and exploration", *Science*, Vol. 153, No. 3731, pp. 25–33, 1966.

5. Hull, C. L., *Principles of behavior*, Vol. 422, Appleton-century-crofts New York, 1943.

6. Freud, S., *An outline of psycho-analysis*, WW Norton & Company, 1969.

7. Freud, S., *Beyond the pleasure principle*, Penguin UK, 2003.

8. Deci, E. L., *Intrinsic Motivation*, Springer US, Boston, MA, 1975, `http://link.springer.com/10.1007/978-1-4613-4446-9`.

9. Baranes, A. and P.-Y. Oudeyer, "Active learning of inverse models with intrinsically motivated goal exploration in robots", *Robotics and Autonomous Systems*, Vol. 61, No. 1, pp. 49–73, 2013.

10. Fournier, P., C. Colas, M. Chetouani and O. Sigaud, "CLIC: Curriculum Learning and Imitation for object Control in non-rewarding environments", *IEEE Transactions on Cognitive and Developmental Systems*, 2019.

11. Czikszentmihalyi, M., *Flow: The psychology of optimal experience*, New York:

Harper & Row, 1990.

12. Schmidhuber, J., "Curious model-building control systems", *Proc. international joint conference on neural networks*, pp. 1458–1463, 1991.

13. Oudeyer, P.-Y., F. Kaplan and V. V. Hafner, "Intrinsic motivation systems for autonomous mental development", *IEEE transactions on evolutionary computation*, Vol. 11, No. 2, pp. 265–286, 2007.

14. Jacobs, R. A., M. I. Jordan, S. J. Nowlan and G. E. Hinton, "Adaptive mixtures of local experts", *Neural computation*, Vol. 3, No. 1, pp. 79–87, 1991.

15. Wolpert, D. M. and M. Kawato, "Multiple paired forward and inverse models for motor control", *Neural networks*, Vol. 11, No. 7-8, pp. 1317–1329, 1998.

16. Kawato, M., "Internal models for motor control and trajectory planning", *Current opinion in neurobiology*, Vol. 9, No. 6, pp. 718–727, 1999.

17. Mohamed, S. and D. J. Rezende, "Variational information maximisation for intrinsically motivated reinforcement learning", *Advances in neural information processing systems*, pp. 2125–2133, 2015.

18. Laversanne-Finot, A., A. Pere and P.-Y. Oudeyer, "Curiosity Driven Exploration of Learned Disentangled Goal Spaces", *Conference on Robot Learning*, pp. 487–504, 2018.

19. Péré, A., S. Forestier, O. Sigaud and P.-Y. Oudeyer, "Unsupervised Learning of Goal Spaces for Intrinsically Motivated Goal Exploration", *International Conference on Learning Representations*, 2018, `https://openreview.net/forum?id=S1DWPP1A-`.

20. Santucci, V. G., G. Baldassarre and M. Mirolli, "GRAIL: a goal-discovering robotic architecture for intrinsically-motivated learning", *IEEE Transactions on*

Cognitive and Developmental Systems, Vol. 8, No. 3, pp. 214–231, 2016.

21. Forestier, S., Y. Mollard and P.-Y. Oudeyer, "Intrinsically motivated goal exploration processes with automatic curriculum learning", *arXiv preprint arXiv:1708.02190*, 2017.

22. Blaes, S., M. V. Pogančić, J. Zhu and G. Martius, "Control What You Can: Intrinsically Motivated Task-Planning Agent", *Advances in Neural Information Processing Systems*, pp. 12520–12531, 2019.

23. Meltzoff, A. N. and M. K. Moore, "Explaining facial imitation: A theoretical model", *Infant and child development*, Vol. 6, No. 3-4, pp. 179–192, 1997.

24. Rolf, M., J. J. Steil and M. Gienger, "Goal babbling permits direct learning of inverse kinematics", *IEEE Transactions on Autonomous Mental Development*, Vol. 2, No. 3, pp. 216–229, 2010.

25. Russell, S. J. and P. Norvig, *Artificial Intelligence: A Modern Approach*, Pearson, 3 edn., 2010.

26. McCulloch, W. S. and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *The bulletin of mathematical biophysics*, Vol. 5, No. 4, pp. 115–133, 1943.

27. Hornik, K., M. Stinchcombe, H. White *et al.*, "Multilayer feedforward networks are universal approximators.", *Neural networks*, Vol. 2, No. 5, pp. 359–366, 1989.

28. Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`.

29. Rumelhart, D. E., G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors", *nature*, Vol. 323, No. 6088, pp. 533–536, 1986.

30. Fukushima, K., "Neocognitron: A self-organizing neural network model for a

mechanism of pattern recognition unaffected by shift in position", *Biological cybernetics*, Vol. 36, No. 4, pp. 193–202, 1980.

31. Maried, E., M. Eldali, O. Ziada and A. Baba, *A Literature Study of Deep learning and its application in Digital Image Processing*, Tech. rep., University of Turkish Aeronautical Association, 2017.

32. Mehta, P., M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher and D. J. Schwab, "A high-bias, low-variance introduction to machine learning for physicists", *Physics reports*, 2019.

33. The MathWorks, Inc., *Convolutional Neural Network*, `https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html`, accessed in May 2020.

34. Weng, L., "From Autoencoder to Beta-VAE", *lilianweng.github.io/lil-log*, 2018, `http://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html`, accessed in May 2020.

35. Kingma, D. P. and M. Welling, "Auto-encoding variational bayes", *arXiv preprint arXiv:1312.6114*, 2013.

36. Universal Robots, *UR10 Website*, `https://www.universal-robots.com/products/ur10-robot/`, accessed in May 2020.

37. Microsoft Corporation, *Kinect Sensor*, `https://docs.microsoft.com/en-us/previous-versions/microsoft-robotics/hh438998(v=msdn.10)?redirectedfrom=MSDN`, accessed in May 2020.

38. roswiki, *Robot Operating System*, `http://wiki.ros.org/`, accessed in May 2020.

39. Rohmer, E., S. P. Singh and M. Freese, "V-REP: A versatile and scalable robot simulation framework", *2013 IEEE/RSJ International Conference on Intelligent*

Robots and Systems, pp. 1321–1326, IEEE, 2013.

40. Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, Vol. 12, pp. 2825–2830, 2011.

41. Chollet, F. *et al.*, "Keras", `https://keras.io`, 2015, accessed in May 2020.

42. Vallat, R., "Pingouin: statistics in Python", *The Journal of Open Source Software*, Vol. 3, No. 31, p. 1026, Nov. 2018.

43. "Statistical functions (scipy.stats)", `https://docs.scipy.org/doc/scipy/reference/stats.html`, accessed in May 2020.

44. "Scikit-posthocs", `https://scikit-posthocs.readthedocs.io/en/latest/`, accessed in May 2020.

45. Deci, E. L. and R. Ryan, "Intrinsic motivation and self-determination in human behavior", *New York: Plenum. doi*, Vol. 10, pp. 978–1, 1985.

46. Blank, D., D. Kumar, L. Meeden and J. B. Marshall, "Bringing up robot: Fundamental mechanisms for creating a self-motivated, self-organizing architecture", *Cybernetics and Systems: An International Journal*, Vol. 36, No. 2, pp. 125–150, 2005.

47. Schmidhuber, J., "Formal theory of creativity, fun, and intrinsic motivation (1990–2010)", *IEEE Transactions on Autonomous Mental Development*, Vol. 2, No. 3, pp. 230–247, 2010.

48. Santucci, V. G., G. Baldassarre and M. Mirolli, "Which is the best intrinsic motivation signal for learning multiple skills?", *Frontiers in neurorobotics*, Vol. 7,

p. 22, 2013.

49. Mirolli, M. and G. Baldassarre, "Functions and mechanisms of intrinsic motivations", *Intrinsically Motivated Learning in Natural and Artificial Systems*, pp. 49–72, Springer, 2013.

50. Baranes, A. and P.-Y. Oudeyer, "R-iac: Robust intrinsically motivated exploration and active learning", *IEEE Transactions on Autonomous Mental Development*, Vol. 1, No. 3, pp. 155–169, 2009.

51. Forestier, S. and P.-Y. Oudeyer, "Modular active curiosity-driven discovery of tool use", *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3965–3972, IEEE, 2016.

52. Chentanez, N., A. G. Barto and S. P. Singh, "Intrinsically motivated reinforcement learning", *Advances in neural information processing systems*, pp. 1281–1288, 2005.

53. Barto, A. G., S. Singh and N. Chentanez, "Intrinsically motivated learning of hierarchical collections of skills", *Proceedings of the 3rd International Conference on Development and Learning*, pp. 112–19, Citeseer, 2004.

54. Uchibe, E. and K. Doya, "Finding intrinsic rewards by embodied evolution and constrained reinforcement learning", *Neural Networks*, Vol. 21, No. 10, pp. 1447–1455, 2008.

55. Hester, T. and P. Stone, "Intrinsically motivated model learning for developing curious robots", *Artificial Intelligence*, Vol. 247, pp. 170–186, 2017.

56. Florensa, C., D. Held, X. Geng and P. Abbeel, "Automatic Goal Generation for Reinforcement Learning Agents", *International Conference on Machine Learning*, pp. 1515–1528, 2018.

57. Eysenbach, B., A. Gupta, J. Ibarz and S. Levine, "Diversity is all you need: Learning skills without a reward function", *arXiv preprint arXiv:1802.06070*, 2018.

58. Colas, C., P.-Y. Oudeyer, O. Sigaud, P. Fournier and M. Chetouani, "CURIOUS: Intrinsically Motivated Modular Multi-Goal Reinforcement Learning", *International Conference on Machine Learning*, pp. 1331–1340, 2019.

59. Barto, A. G. and S. Mahadevan, "Recent advances in hierarchical reinforcement learning", *Discrete event dynamic systems*, Vol. 13, No. 1-2, pp. 41–77, 2003.

60. Kulkarni, T. D., K. Narasimhan, A. Saeedi and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation", *Advances in neural information processing systems*, pp. 3675–3683, 2016.

61. Vezhnevets, A. S., S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning", *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3540–3549, JMLR. org, 2017.

62. Nachum, O., S. S. Gu, H. Lee and S. Levine, "Data-efficient hierarchical reinforcement learning", *Advances in Neural Information Processing Systems*, pp. 3303–3313, 2018.

63. Levy, A., G. Konidaris, R. Platt and K. Saenko, "Learning multi-level hierarchies with hindsight", *Proceedings of International Conference on Learning Representations*, 2019.

64. Haber, N., D. Mrowca, L. Fei-Fei and D. L. Yamins, "Emergence of structured behaviors from curiosity-based intrinsic motivation", *arXiv preprint arXiv:1802.07461*, 2018.

65. Mannella, F., V. G. Santucci, E. Somogyi, L. Jacquey, K. J. O'Regan and G. Bal-

dassarre, "Know your body through intrinsic goals", *Frontiers in neurorobotics*, Vol. 12, p. 30, 2018.

66. Nagai, Y., "Predictive learning: its key role in early cognitive development", *Philosophical Transactions of the Royal Society B*, Vol. 374, No. 1771, p. 20180030, 2019.

67. Gaussier, P. and A. Pitti, "Reaching and Grasping: what we can learn from psychology and robotics", *hal-01609659*, 2017.

68. Bugur, S., E. Oztop, Y. Nagai and E. Ugur, "Effect regulated projection of robot's action space for production and prediction of manipulation primitives through learning progress and predictability based exploration", *IEEE Transactions on Cognitive and Developmental Systems*, 2019.

69. Ivaldi, S., N. Lyubova, A. Droniou, D. Gerardeaux-Viret, D. Filliat, V. Padois, O. Sigaud, P.-Y. Oudeyer *et al.*, "Learning to recognize objects through curiosity-driven manipulation with the iCub humanoid robot", *2013 IEEE Third Joint International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pp. 1–8, IEEE, 2013.

70. Oudeyer, P.-Y. *et al.*, "Socially guided intrinsic motivation for robot learning of motor skills", *Autonomous Robots*, Vol. 36, No. 3, pp. 273–294, 2014.

71. Duminy, N., S. M. Nguyen and D. Duhaut, "Learning a set of interrelated tasks by using a succession of motor policies for a socially guided intrinsically motivated learner", *Frontiers in neurorobotics*, Vol. 12, p. 87, 2019.

72. Ugur, E., M. R. Dogar, M. Cakmak and E. Sahin, "Curiosity-driven learning of traversability affordance on a mobile robot", *2007 IEEE 6th International Conference on Development and Learning*, pp. 13–18, IEEE, 2007.

73. Ugur, E. and J. Piater, "Emergent structuring of interdependent affordance learn-

ing tasks using intrinsic motivation and empirical feature selection", *IEEE Transactions on Cognitive and Developmental Systems*, Vol. 9, No. 4, pp. 328–340, 2016.

74. Baldassarre, G., W. Lord, G. Granato and V. G. Santucci, "An embodied agent learning affordances with intrinsic motivations and solving extrinsic tasks with attention and one-step planning", *Frontiers in neurorobotics*, Vol. 13, p. 45, 2019.

75. Bengio, Y., A. Courville and P. Vincent, "Representation learning: A review and new perspectives", *IEEE transactions on pattern analysis and machine intelligence*, Vol. 35, No. 8, pp. 1798–1828, 2013.

76. Bowling, M., A. Ghodsi and D. Wilkinson, "Action respecting embedding", *Proceedings of the 22nd international conference on Machine learning*, pp. 65–72, 2005.

77. Boots, B., S. M. Siddiqi and G. J. Gordon, "Closing the learning-planning loop with predictive state representations", *The International Journal of Robotics Research*, Vol. 30, No. 7, pp. 954–966, 2011.

78. Lange, S., M. Riedmiller and A. Voigtländer, "Autonomous reinforcement learning on raw visual input data in a real world application", *The 2012 international joint conference on neural networks (IJCNN)*, pp. 1–8, IEEE, 2012.

79. Jonschkowski, R. and O. Brock, "State Representation Learning in Robotics: Using Prior Knowledge about Physical Interaction.", *Robotics: Science and Systems*, 2014.

80. Jonschkowski, R. and O. Brock, "Learning state representations with robotic priors", *Autonomous Robots*, Vol. 39, No. 3, pp. 407–428, 2015.

81. Watter, M., J. Springenberg, J. Boedecker and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images", *Advances in neural information processing systems*, pp. 2746–2754, 2015.

82. Banijamali, E., R. Shu, H. Bui, A. Ghodsi *et al.*, "Robust Locally-Linear Controllable Embedding", *International Conference on Artificial Intelligence and Statistics*, pp. 1751–1759, 2018.

83. Machado, M. C., C. Rosenbaum, X. Guo, M. Liu, G. Tesauro and M. Campbell, "Eigenoption discovery through the deep successor representation", *arXiv preprint arXiv:1710.11089*, 2017.

84. Sutton, R. S. and A. G. Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

85. Dayan, P., "Improving generalization for temporal difference learning: The successor representation", *Neural Computation*, Vol. 5, No. 4, pp. 613–624, 1993.

86. Whitney, W., R. Agarwal, K. Cho and A. Gupta, "Dynamics-Aware Embeddings", *International Conference on Learning Representations*, 2020, `https://openreview.net/forum?id=BJgZGeHFPH`.

87. Blau, T., L. Ott and F. Ramos, "Bayesian Curiosity for Efficient Exploration in Reinforcement Learning", *arXiv preprint arXiv:1911.08701*, 2019.

88. Hafez, M. B., C. Weber, M. Kerzel and S. Wermter, "Efficient Intrinsically Motivated Robotic Grasping with Learning-Adaptive Imagination in Latent Space", *2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pp. 1–7, IEEE, 2019.

89. Zhao, R., S. Tiomkin and P. Abbeel, "Learning Efficient Representation for Intrinsic Motivation", *arXiv preprint arXiv:1912.02624*, 2019.

90. Weng, J., J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur and E. Thelen, "Autonomous mental development by robots and animals", *Science*, Vol. 291, No. 5504, pp. 599–600, 2001.

91. Ryan, R. M. and E. L. Deci, "Intrinsic and extrinsic motivations: Classic definitions and new directions", *Contemporary educational psychology*, Vol. 25, No. 1, pp. 54–67, 2000.

92. Sperati, V. and G. Baldassarre, "Learning where to look with movement-based intrinsic motivations: A bio-inspired model", *4th International Conference on Development and Learning and on Epigenetic Robotics*, pp. 461–468, IEEE, 2014.

93. Altman, N. S., "An introduction to kernel and nearest-neighbor nonparametric regression", *The American Statistician*, Vol. 46, No. 3, pp. 175–185, 1992.

94. Kingma, D. P. and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

95. Zeiler, M. D., "Adadelta: an adaptive learning rate method", *arXiv preprint arXiv:1212.5701*, 2012.

96. Gerber, R. J., T. Wilks and C. Erdie-Lalena, "Developmental milestones: motor development", *Pediatrics in Review*, Vol. 31, No. 7, pp. 267–277, 2010.

97. Redgrave, P., T. J. Prescott and K. Gurney, "The basal ganglia: a vertebrate solution to the selection problem?", *Neuroscience*, Vol. 89, No. 4, pp. 1009–1023, 1999.

98. Mirolli, M., V. G. Santucci and G. Baldassarre, "Phasic dopamine as a prediction error of intrinsic and extrinsic reinforcements driving both action acquisition and reward maximization: A simulated robotic study", *Neural Networks*, Vol. 39, pp. 40–51, 2013.

99. McLeod, S., "Jean Piaget's Theory and Stages of Cognitive Development", *Simply Psychology*, 2018, `https://www.simplypsychology.org/piaget.html`, accessed in May 2020.

100. Piaget, J. and M. Cook, *The origins of intelligence in children*, Vol. 8, International Universities Press New York, 1952.

101. Piaget, J. and B. Inhelder, *The Psychology of the Child*, Trans. Helen Weaver. New York: Basic Books, 1969.